Computer Vision

Deep Generative Models

VAEs, GANs, Diffusion Models

Luis F. Teixeira | M.EIC | FEUP

Can we generate images from scratch?

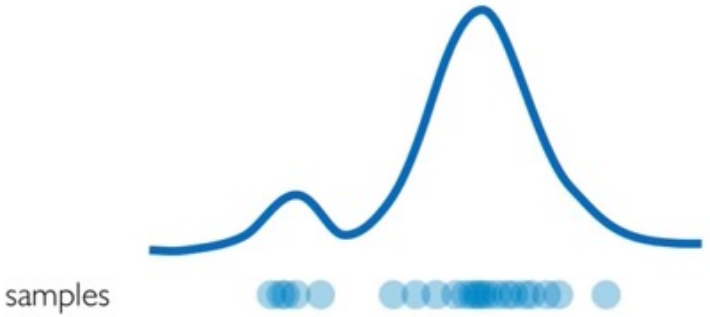


Which one is computer-generated?

Generative modeling

Goal: Take as input training samples from some distribution and **learn a model that represents that distribution**

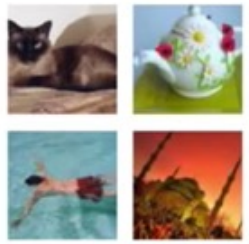
density estimation



sample estimation

Input samples

Training data ~ $P_{data}(x)$



Generated samples

Generated ~ $P_{model}(x)$

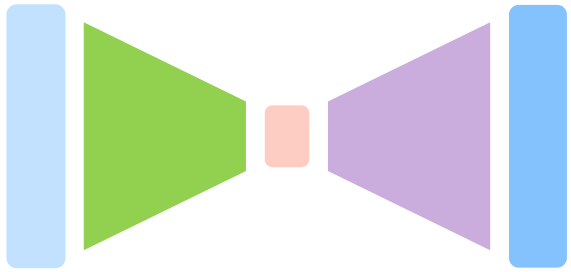
How to generate $P_{model}(x)$ similar to $P_{data}(x)$?

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Latent variable models

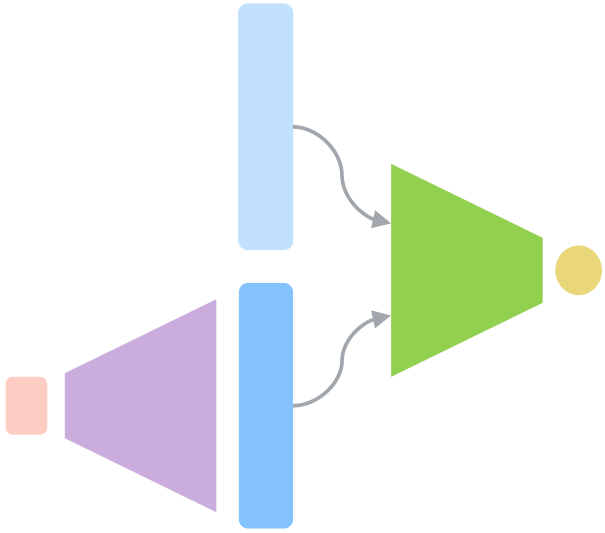
Autoencoders and Variational Autoencoders (VAEs)

Learn **lower-dimensional** latent space and **sample** to generate input reconstructions



Generative Adversarial Networks (GANs)

Competing **generator** and **discriminator** networks



adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

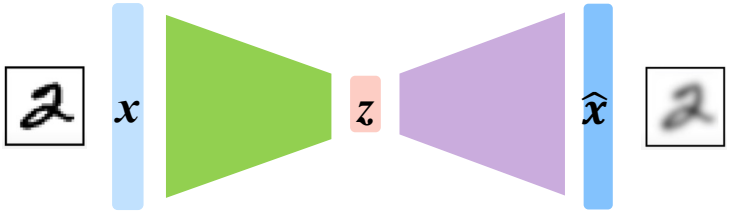
Latent variable models

- **Latent variables** are variables that we can not observe/measure directly but that are responsible for the observable variables
- We want to **learn these latent variables from the observable data**
  - NNs are well suited for this problem because of their capability of approximating complex data distributions

Autoencoders

Bottleneck hidden layer forces the network to learn a compressed latent representation

Reconstruction loss forces the latent space to capture (or encode) as much "information" about the data as possible?

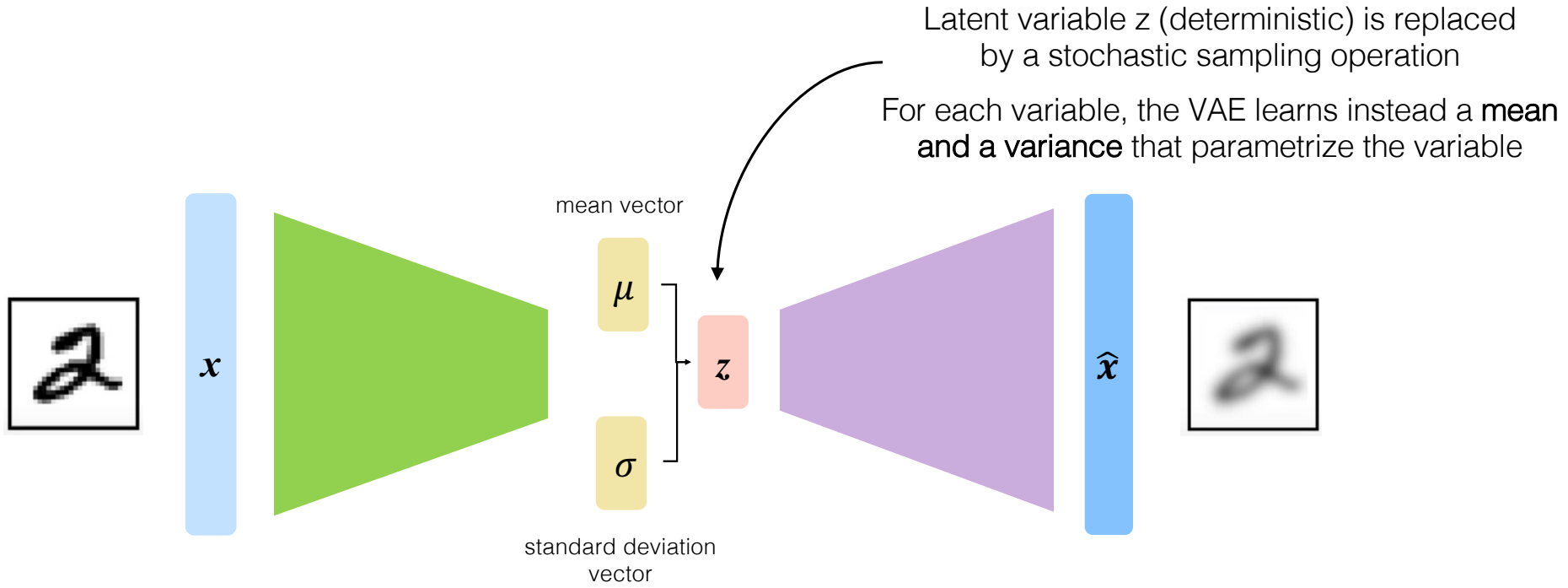


The learned representation is deterministic.

Is that enough for image generation?

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

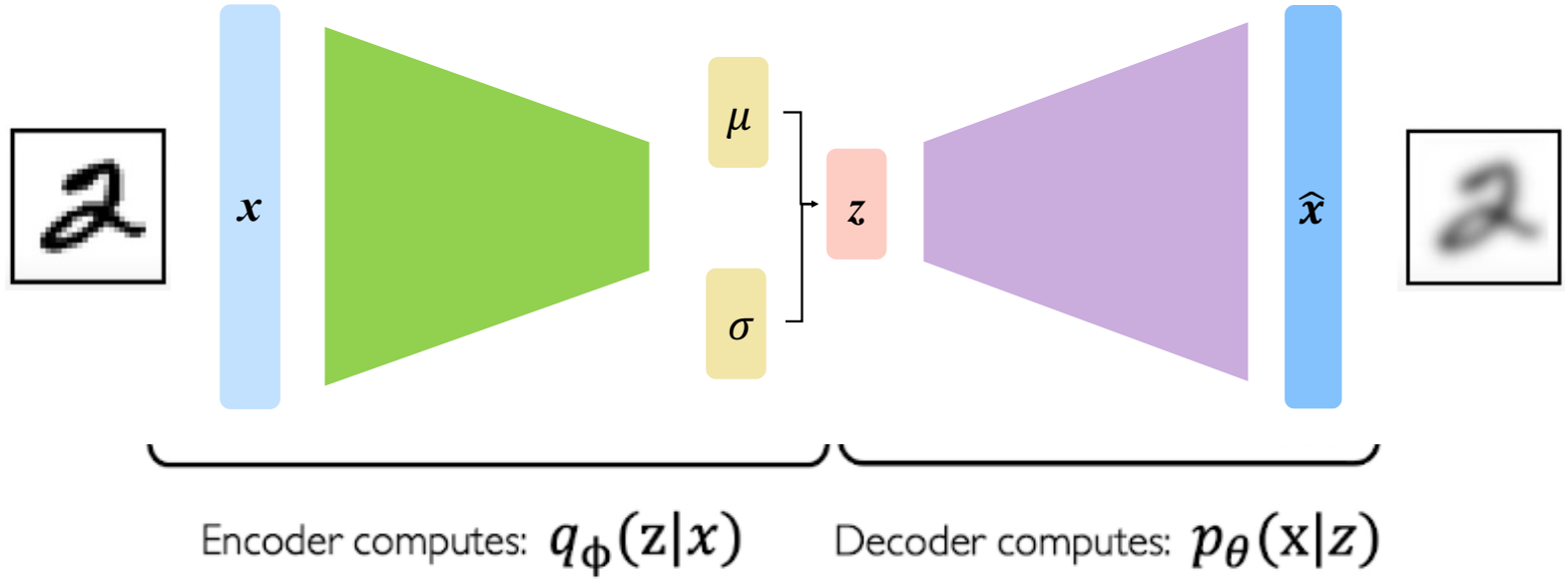
Variational Autoencoders (VAEs)



After training, use the generator to create new examples

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

VAE optimization



Encoder computes: $q_\phi(\mathbf{z}|x)$ Decoder computes: $p_\theta(\mathbf{x}|z)$

$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

$$\|x - \hat{x}\|^2 \qquad D(q_\phi(z|x) \,||\, p(z))$$

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

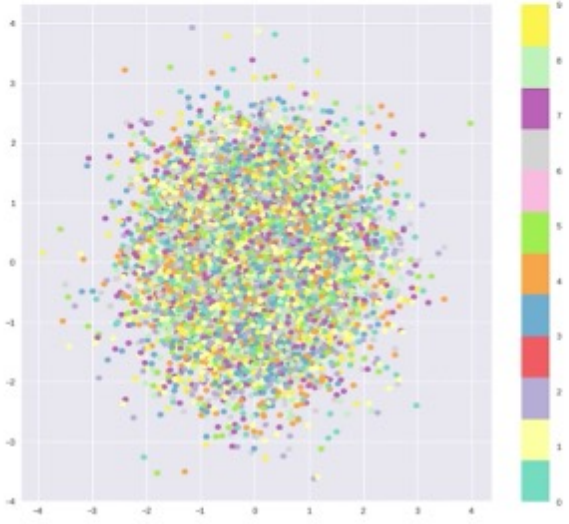
VAE optimization

$$D\left(q_\phi(z|x) \parallel p(z)\right)$$

Inferred latent distribution

Fixed prior on latent distribution

minimize the **divergence** between the **learned latent distribution** and a **specified prior** (initial hypothesis about the distribution of z) → encourage the network to learn a distribution similar to the prior



Common choice of prior – Normal Gaussian:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

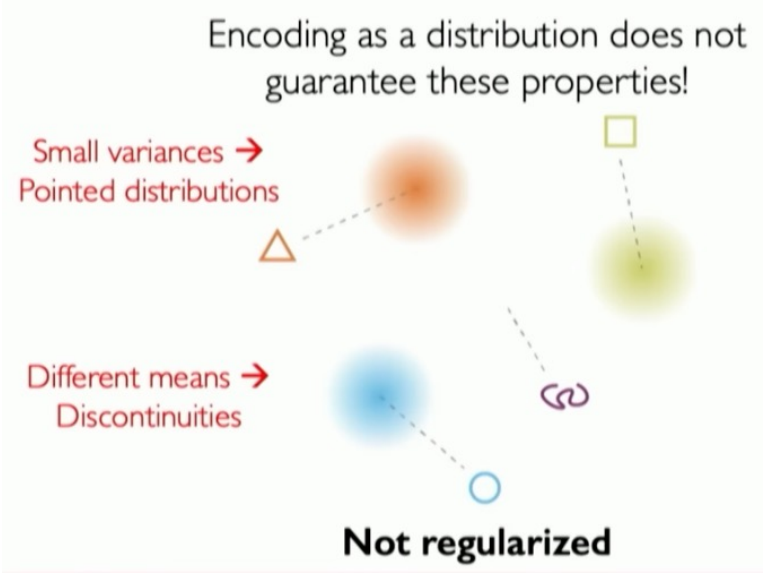
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (i.e., by memorizing the data)

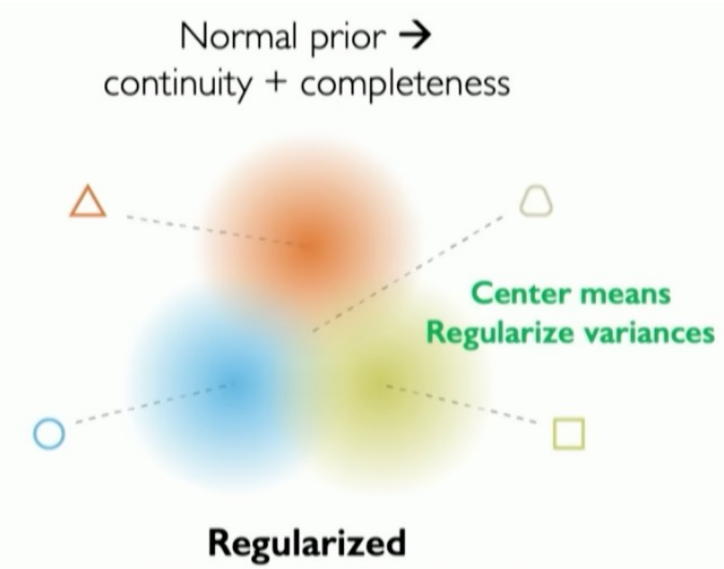
adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Regularization + Normal Prior

What properties do we want to achieve from regularization?

- **Continuity**: points are close in latent space → similar content after decoding
- **Completeness**: sampling from latent space → "meaningful" content after decoding



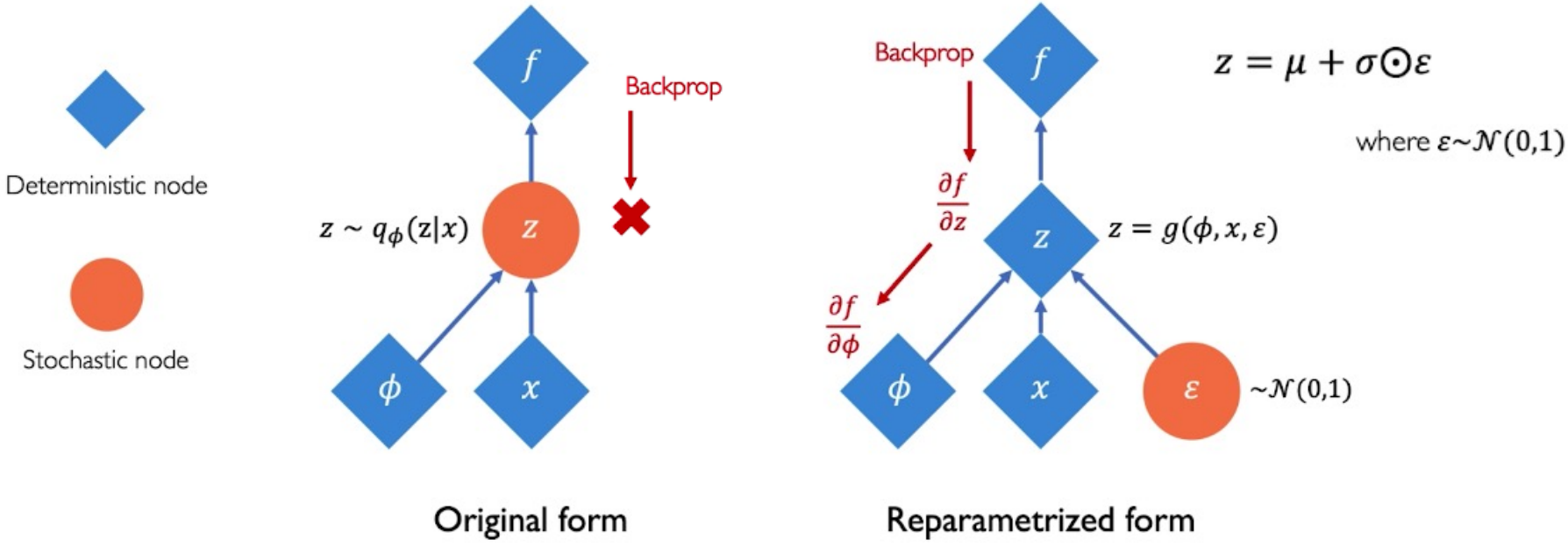


Trade-off: the more we regularize, the higher the risk the quality of the reconstruction will suffer

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

VAE optimization

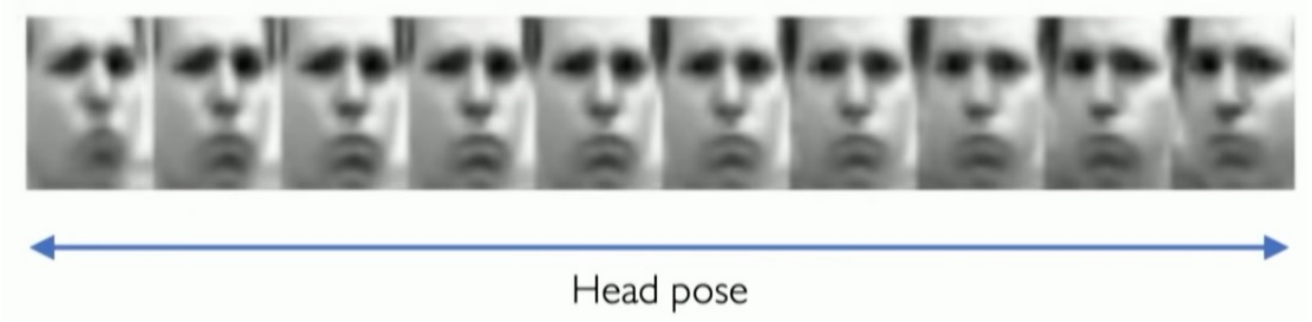
Sampling layer is **not differentiable** → solution: reparametrize the sampling layer



adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

VAE latent variables

Slowly increase or decrease a **single latent variable**
Keep all other variables fixed



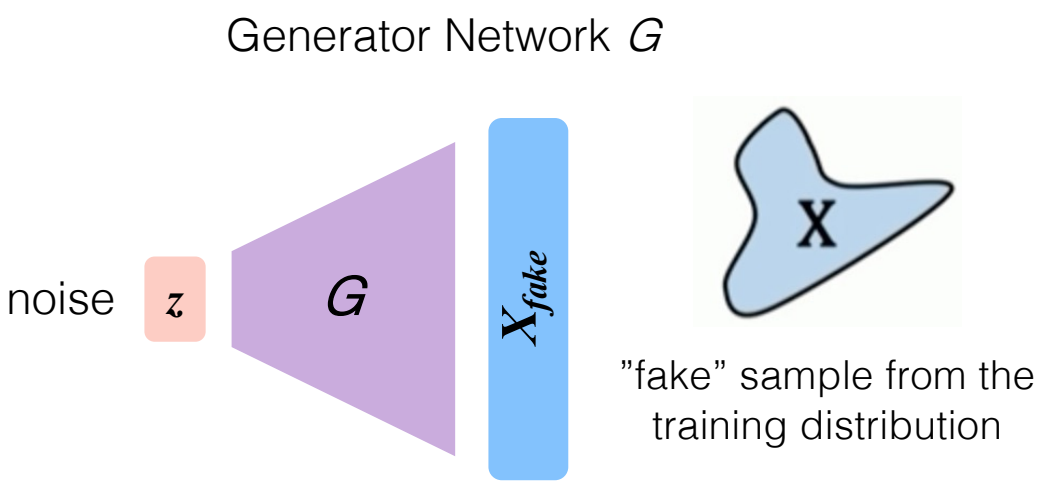
Different dimensions of z encodes **different interpretable latent features**

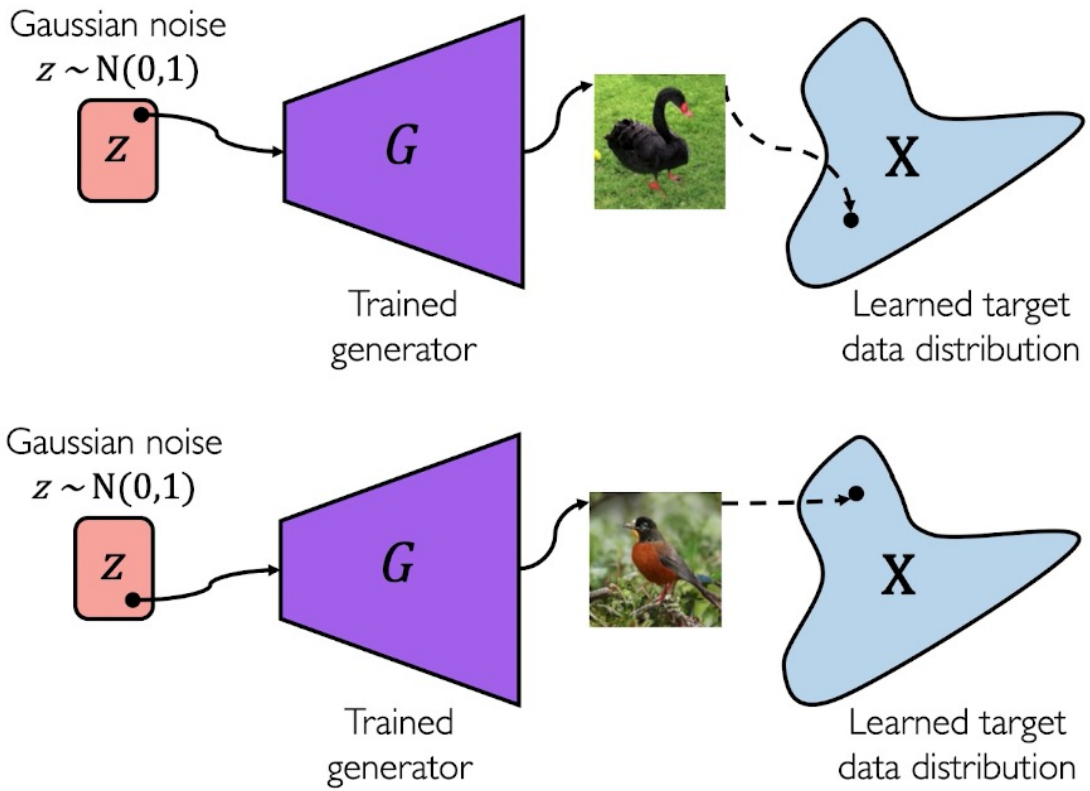
What if we just want to sample?

Idea: don't explicitly model density, and instead just sample to generate new instances

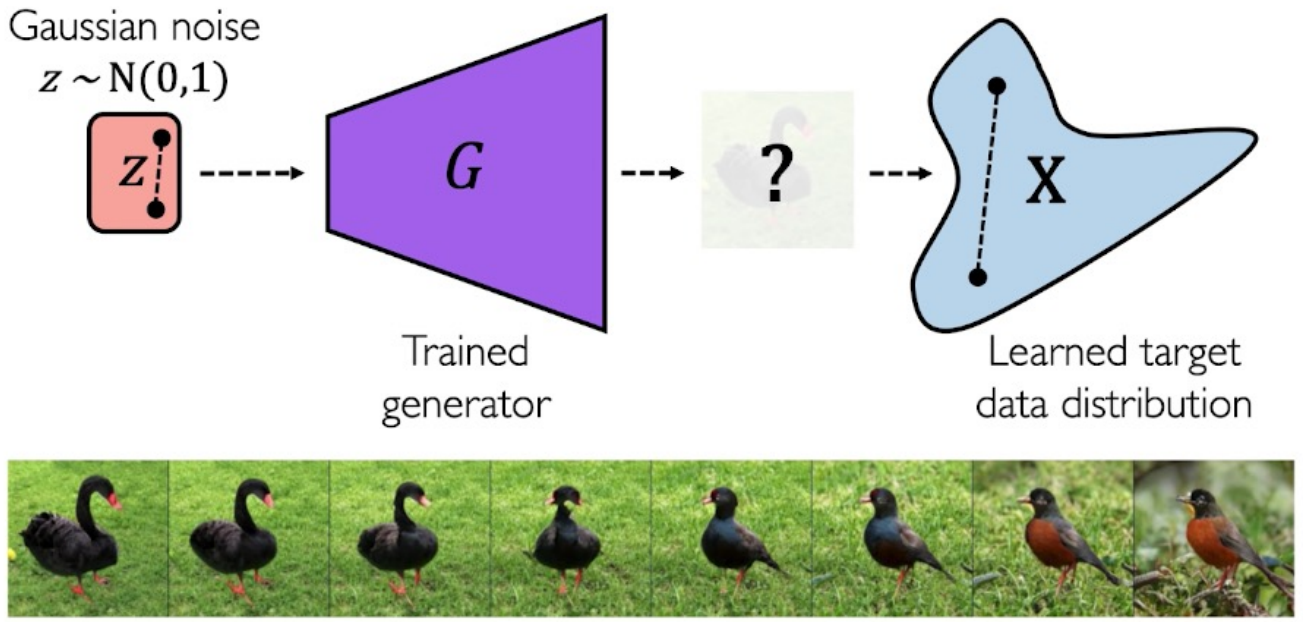
Problem: want to sample from complex distributions – can't do this directly

Solution: sample from something simple (noise), learn a transformation to the training distribution





adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

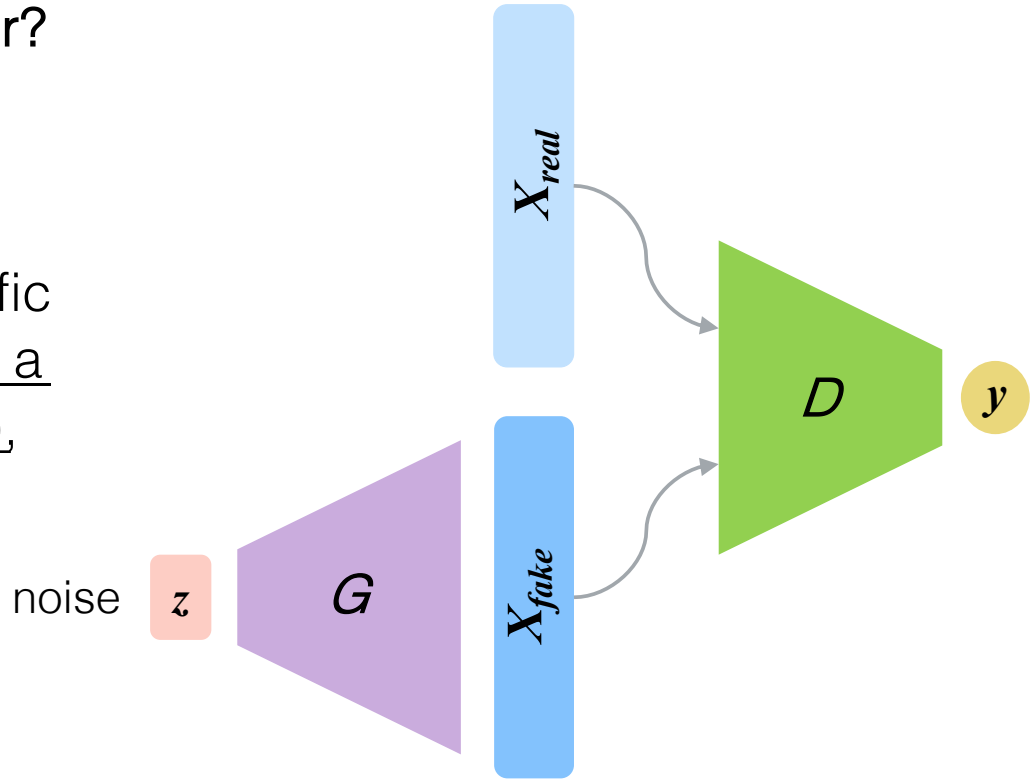


adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Generative Adversarial Networks (GANs)

How can we train the generator?

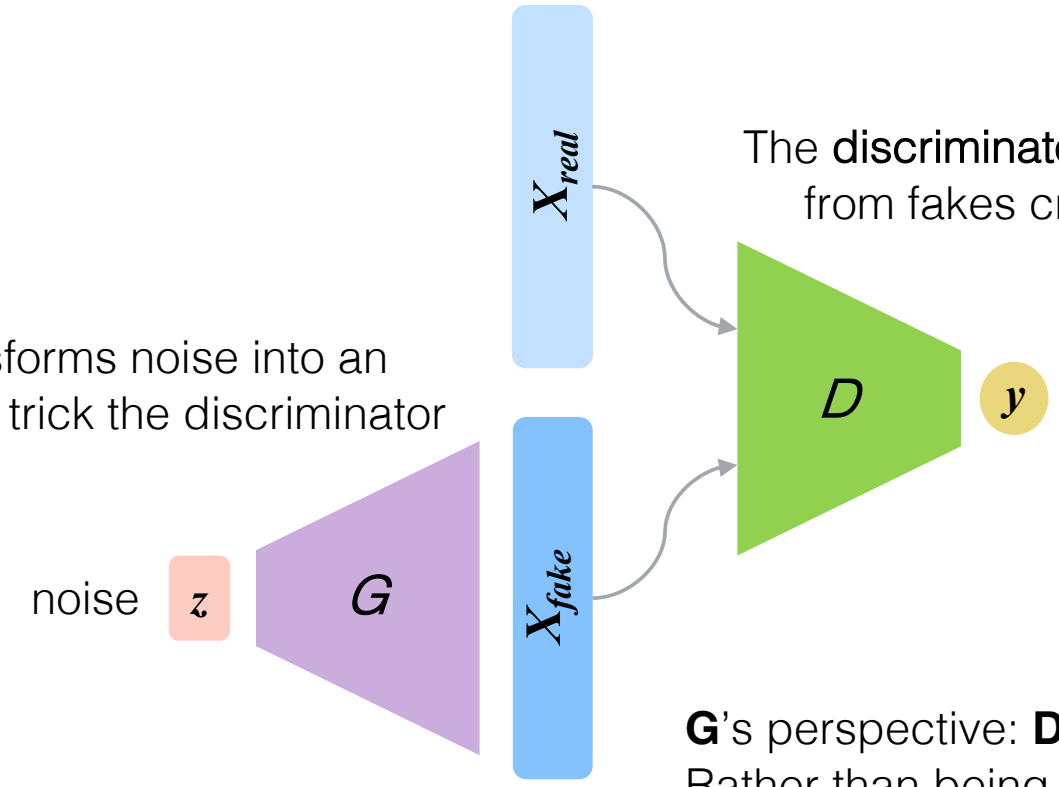
Adversarial Training
the generator is not trained to minimize a distance to a specific input or distribution, but to fool a discriminator (binary classifier), that at the same time learns to discriminate better



Generative Adversarial Networks
Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014

Generative Adversarial Networks (GANs)

The **generator** transforms noise into an imitation of the data to trick the discriminator

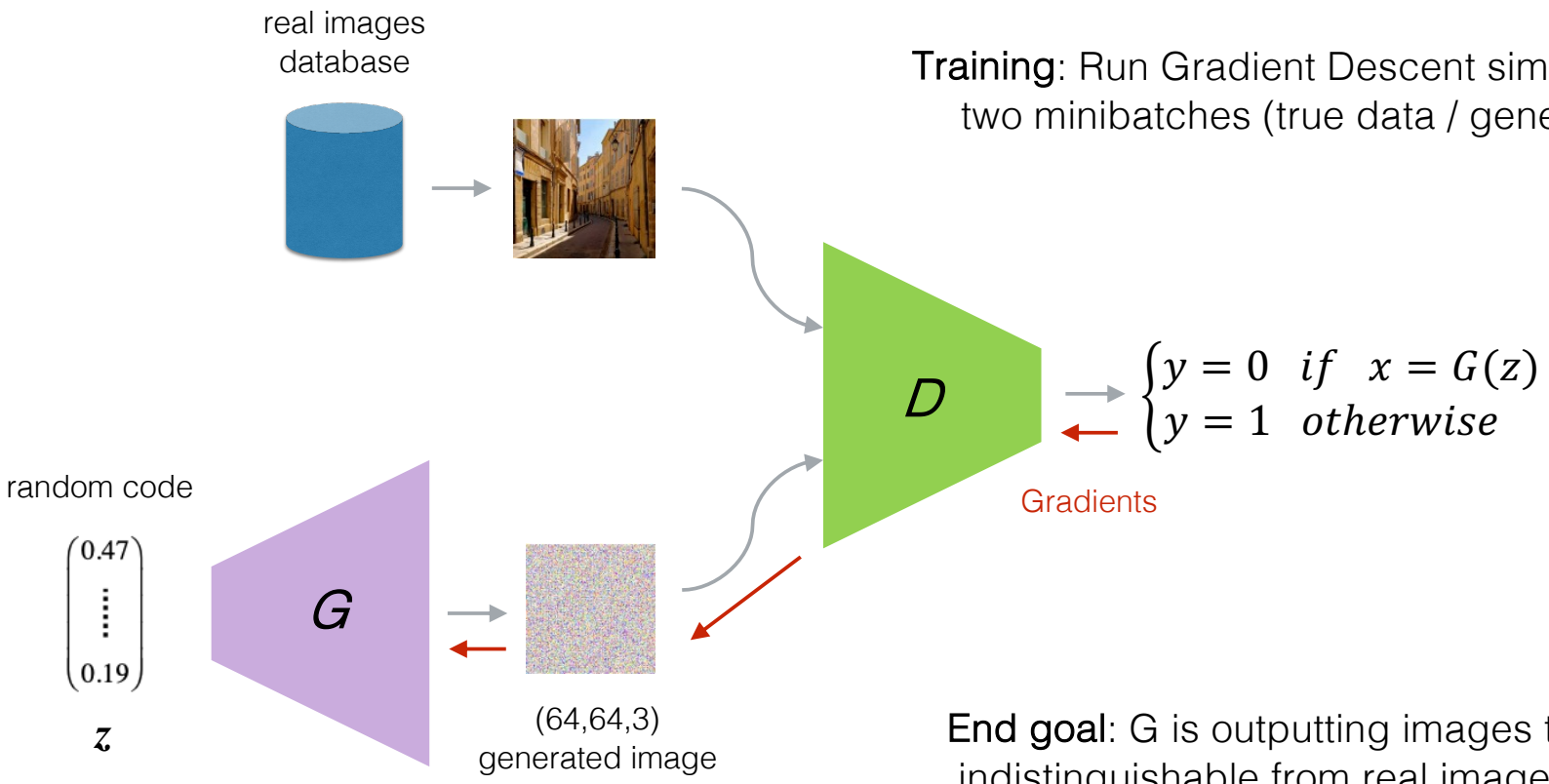


The **discriminator** tries to identify real data from fakes created by the generator

G's perspective: **D** is a loss function.
Rather than being hand-designed, it is *learned*.

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Generative Adversarial Networks (GANs)



Training: Run Gradient Descent simultaneously on two minibatches (true data / generated data)

End goal: G is outputting images that are indistinguishable from real images for D

Generative Adversarial Networks
Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014

Generative Adversarial Networks (GANs)

Training procedure, we want to minimize:

Labels: $\begin{cases} y_{real} \ \ is\ always\ 1 \\ y_{gen} \ \ is\ always\ 0 \end{cases}$

The cost of the **discriminator**

$$\mathcal{L}_D = \underbrace{-\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} y_{real}^{(i)} . \log(D(x^{(i)}))}_{\text{cross-entropy 1: "D should correctly label real data as 1"}} \underbrace{-\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} (1 - y_{gen}^{(i)}) . \log(1 - D(G(z^{(i)})))}_{\text{cross-entropy 2: "D should correctly label generated data as 0"}}$$

The cost of the **generator**

$$\mathcal{L}_G = -\mathcal{L}_D = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(1 - D(G(z^{(i)})))$$

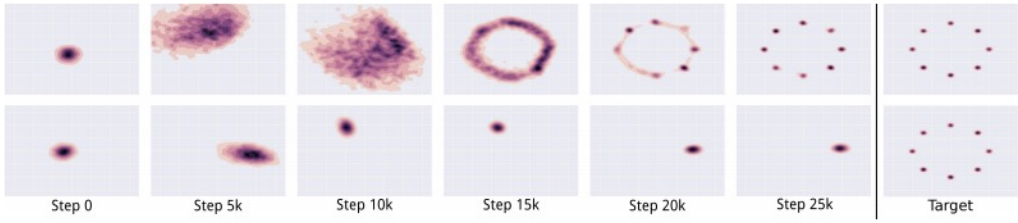
"G should try to fool D: by minimizing the opposite of what D is trying to minimize"

GANs Difficulties

Mode collapse

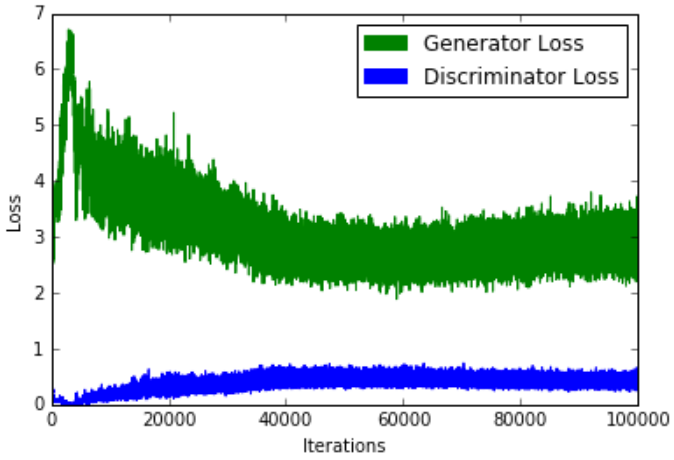
- Natural data distributions are highly complex and multimodal (many modes)
- Each **mode** represents a concentration of **similar data samples**
- During mode collapse, the generator produces samples that belong to a **limited set of modes**

mode collapse →



Convergence

- When to stop the training process?
- Since the generator loss improves when the discriminator loss degrades (and vice-versa)



Quality metric

- How to quantitatively evaluate the quality of the generated samples?

Metrics

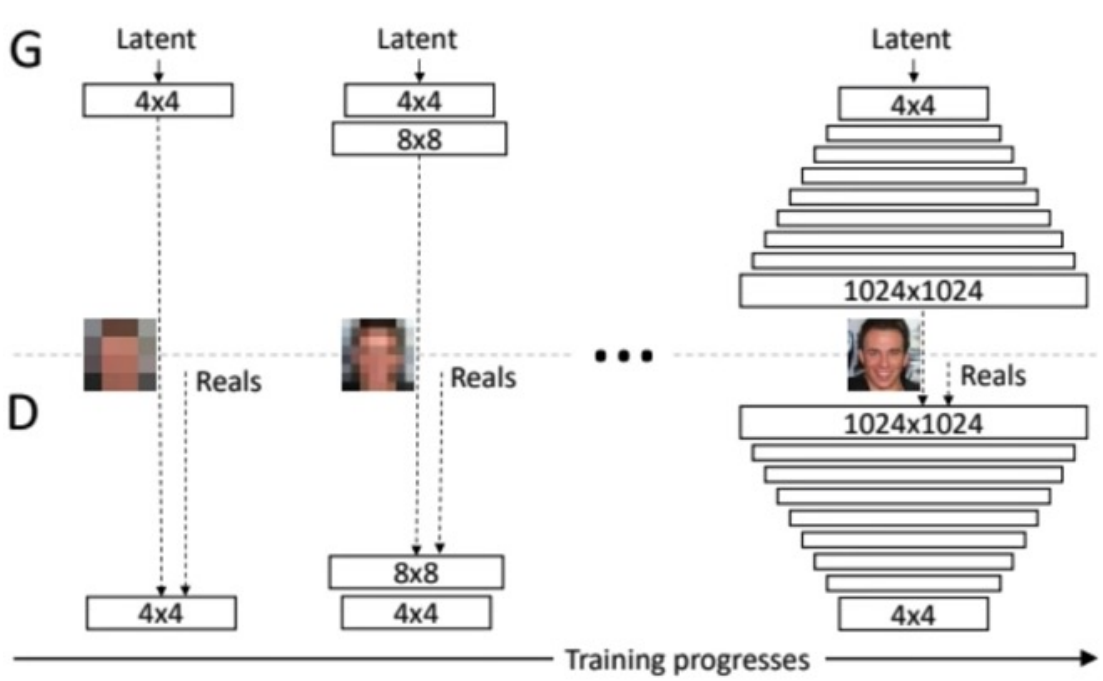
Inception Score (IS)

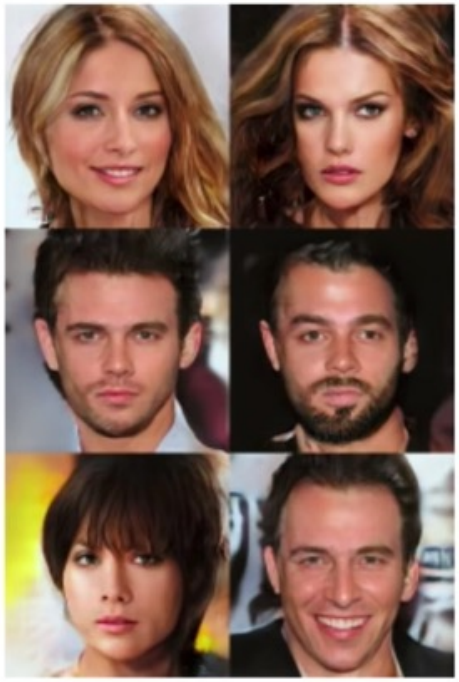
- Uses an Inception-v3 Network, pretrained on the ImageNet dataset to measure two properties: **quality** and **diversity**
- A **higher IS is better**
- A drawback is that **statistics of the real data are not compared with the statistics of the generated data** → Fréchet distance resolves this by **comparing the mean and covariance** of the real and generated images

Fréchet Inception Distance (FID)

- Fréchet Inception Distance (FID) performs the same analysis, but on the **feature maps** produced by passing the real and generated images through a pre-trained Inception-v3 Network
- A **lower FID score is better**, as it explains that the statistics of the generated images are very similar to that of the real images.

Progressive growing of GANs





Progressive GAN [Karras et al., 2018]

https://www.youtube.com/watch?v=kSLJriaOumA

https://www.youtube.com/watch?v=c-NJtV9Jvp0

https://thispersondoesnotexist.com/

Style GAN [Karras et al., 2018]

Style GAN 2 [Karras et al., 2019]

Image-to-Image Translation

Input $\mathbf{x}$

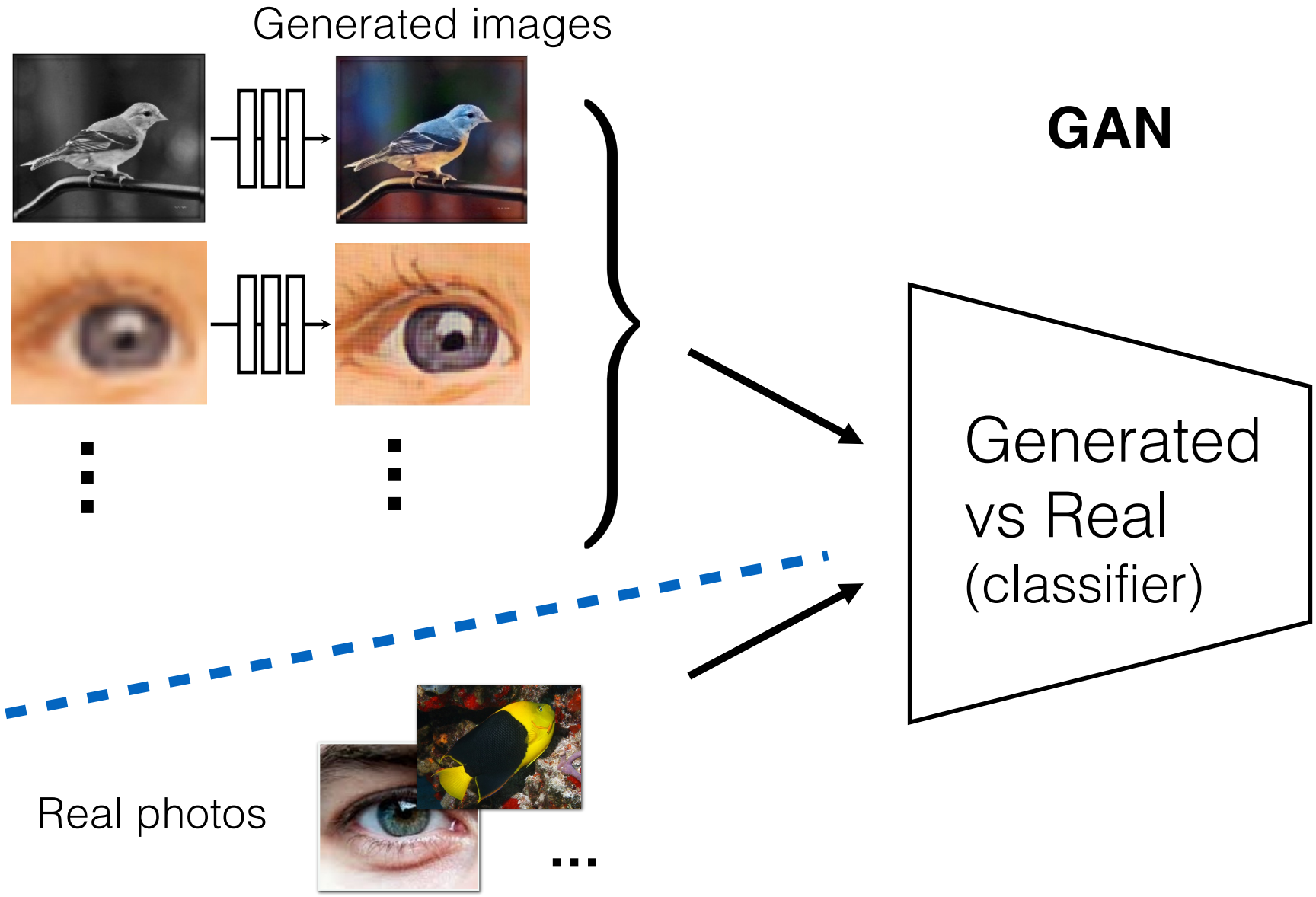
Output $\mathbf{y}$

$\mathcal{F}$

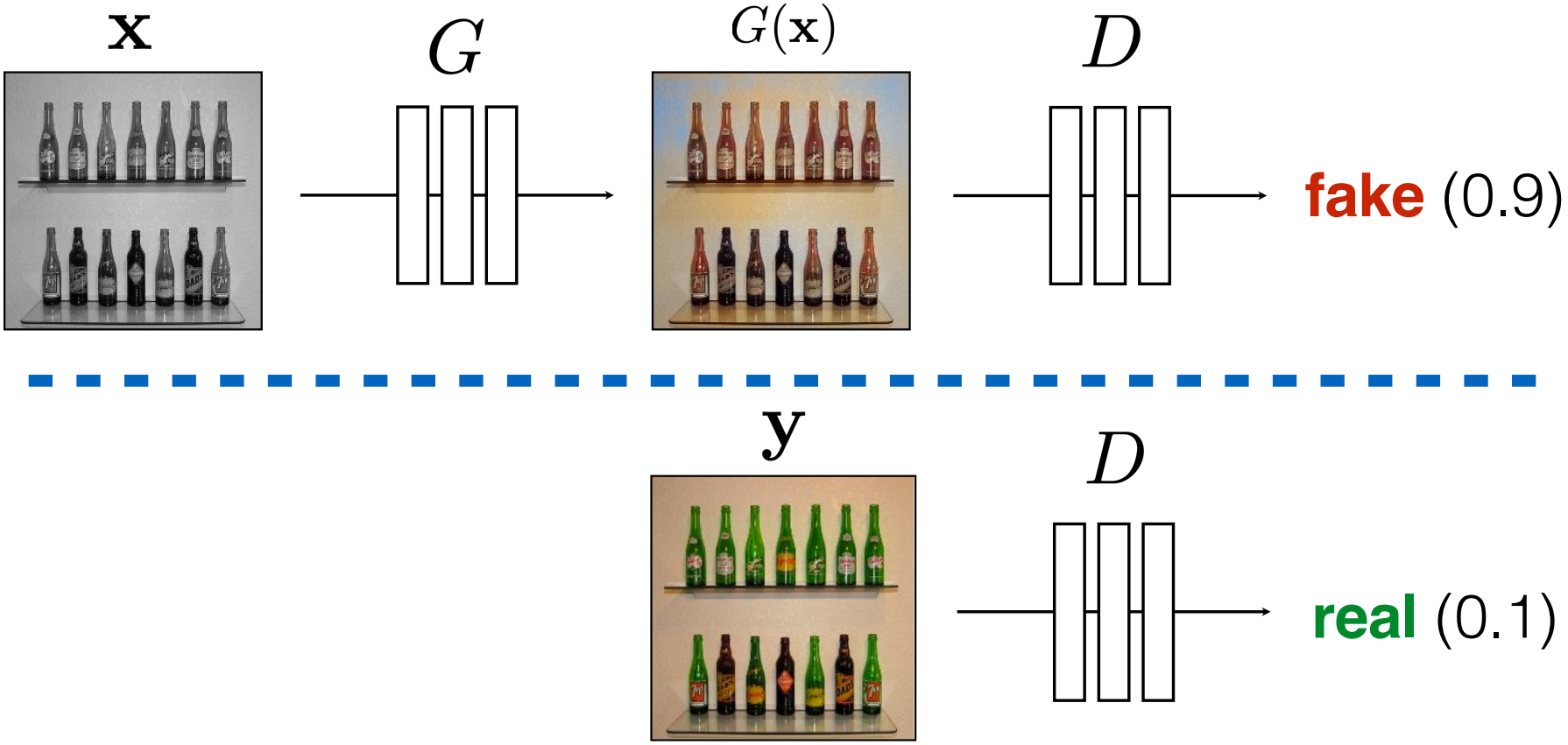
Training data

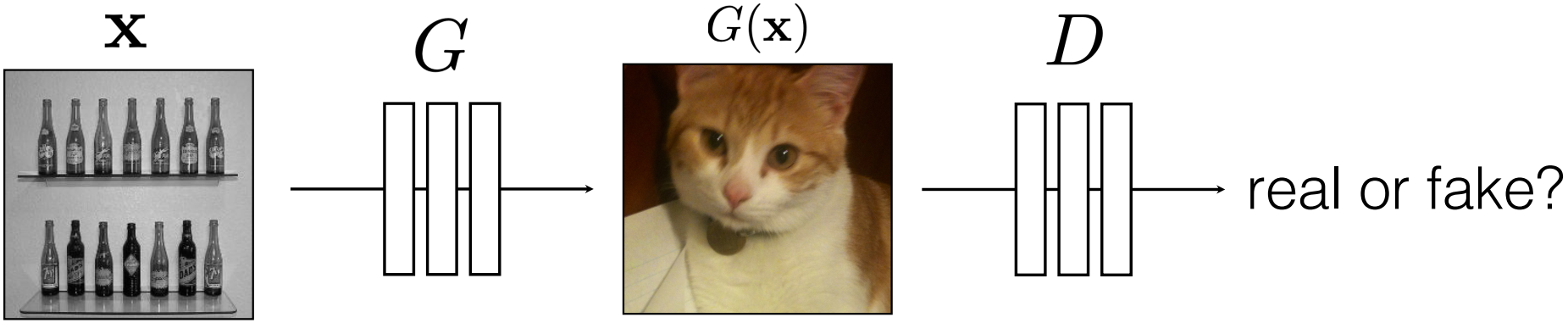
$\mathbf{x}$ $\mathbf{y}$

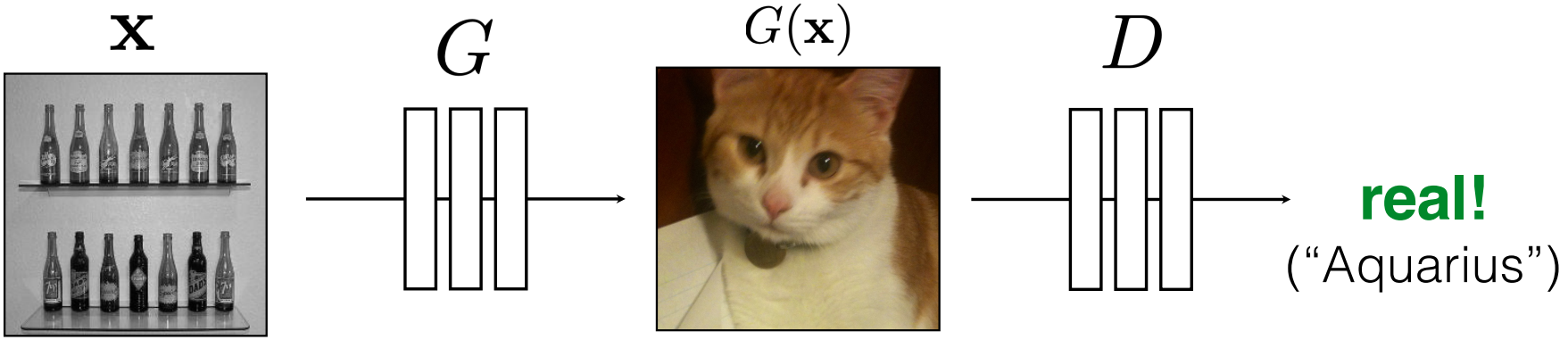
Without explicitly defining a loss function?

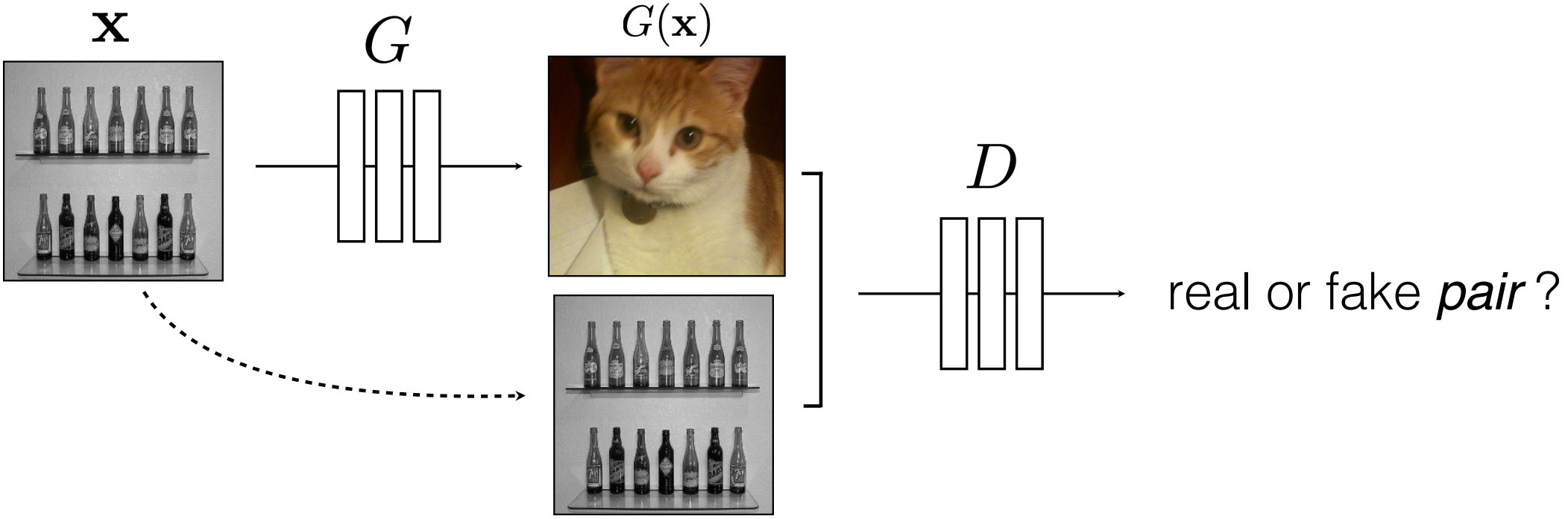


adapted from MIT 6.819/6.869 - Bill Freeman, Antonio Torralba, Phillip Isola

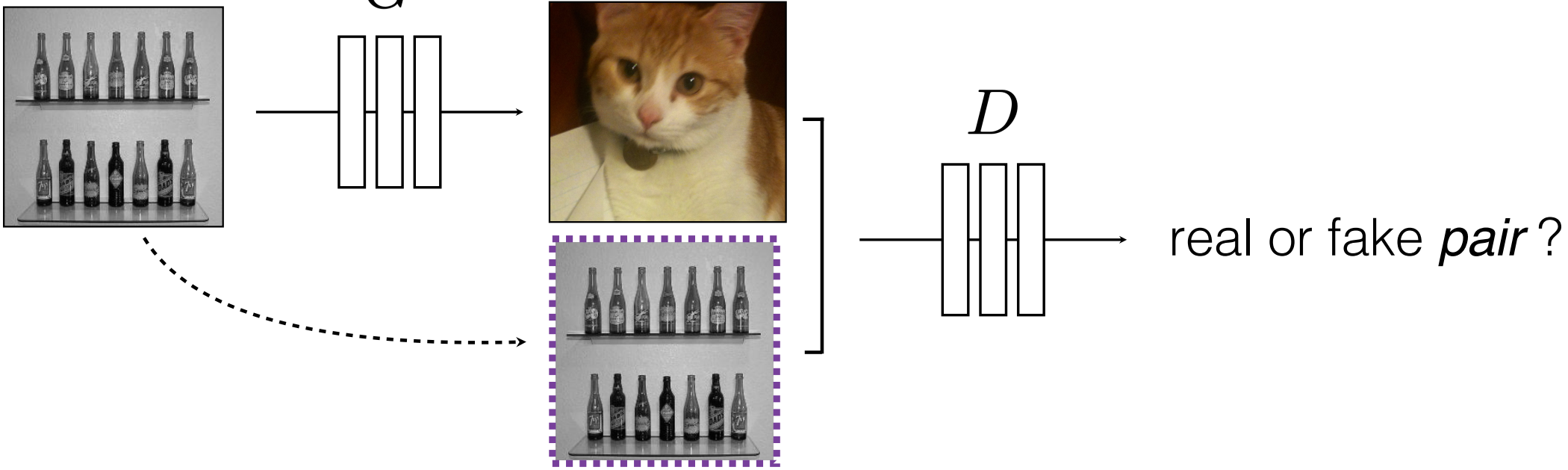

x
G
G(x)
D
fake (0.9)
y
D
real (0.1)


$\mathbf{x}$
G
$G(\mathbf{x})$
D
real or fake?

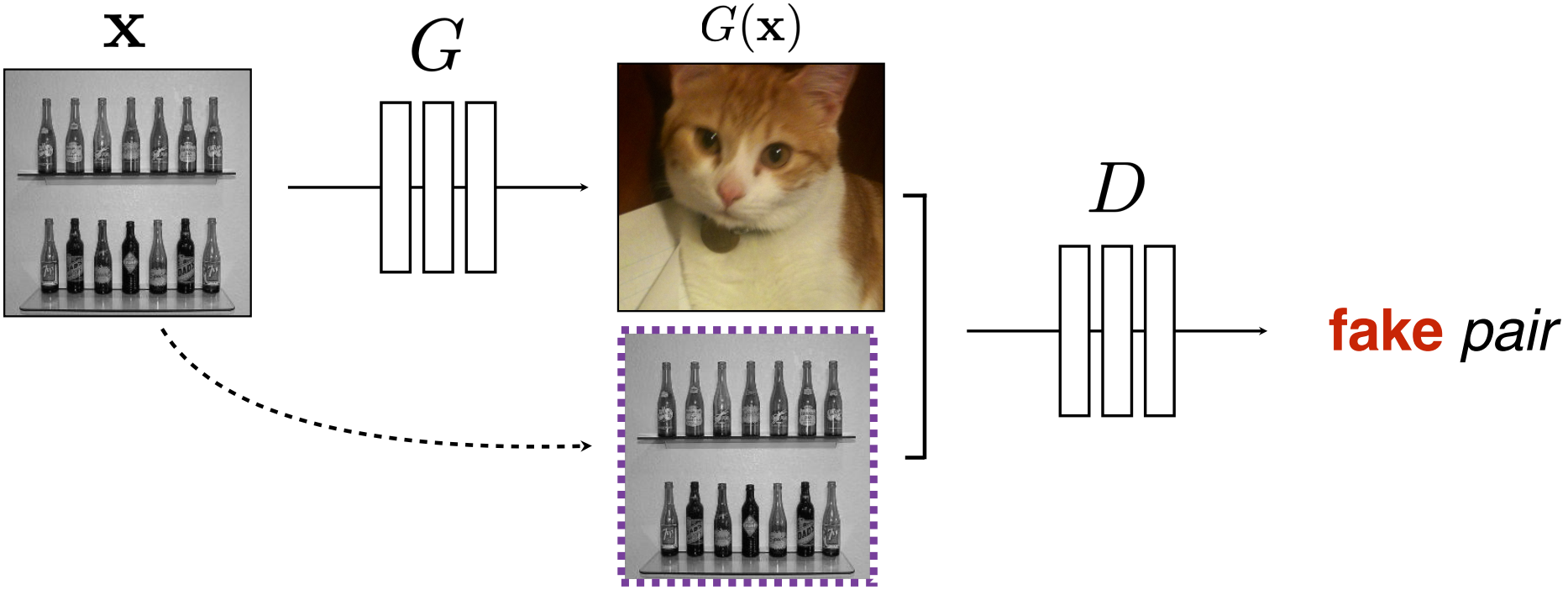

$\mathbf{x}$
G
$G(\mathbf{x})$
D
real!
("Aquarius")



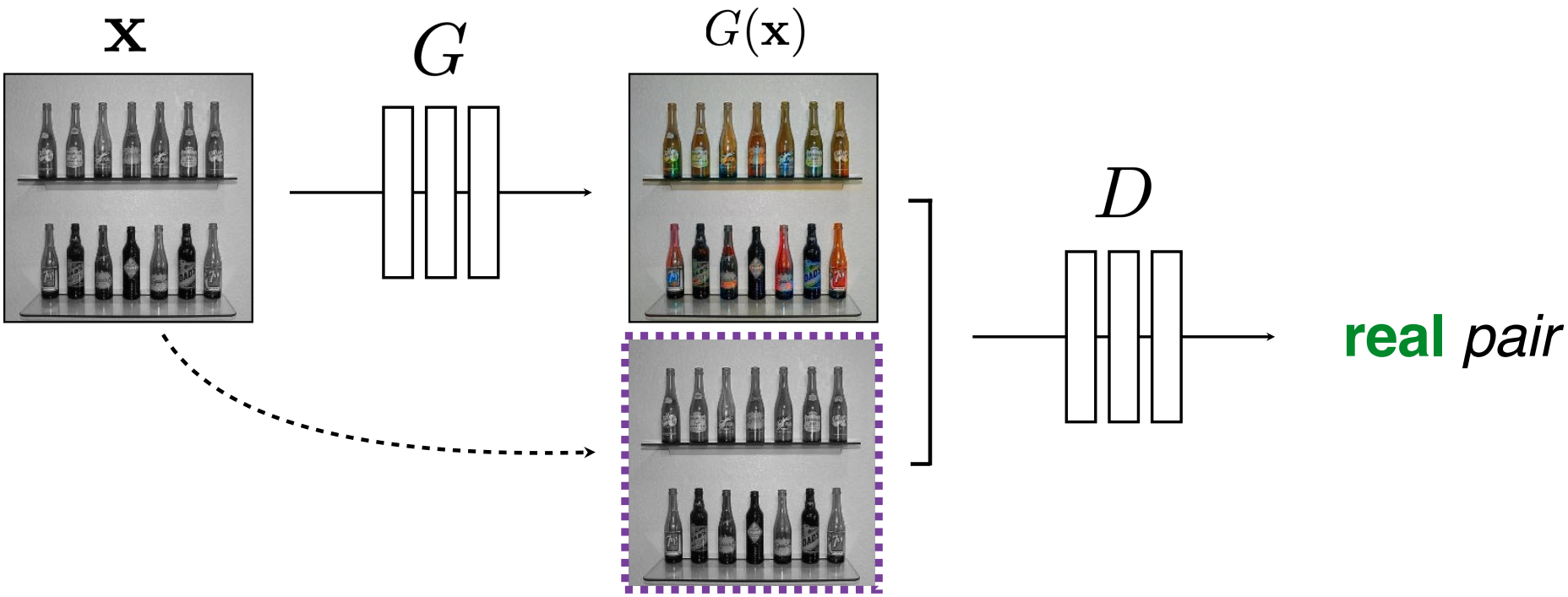
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

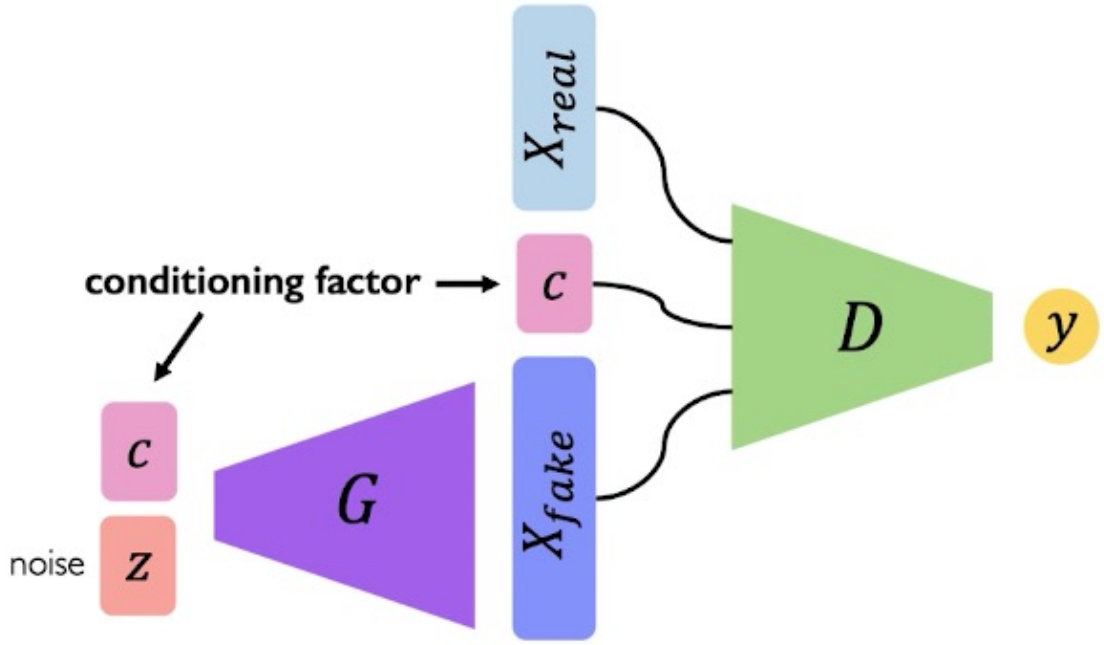


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$



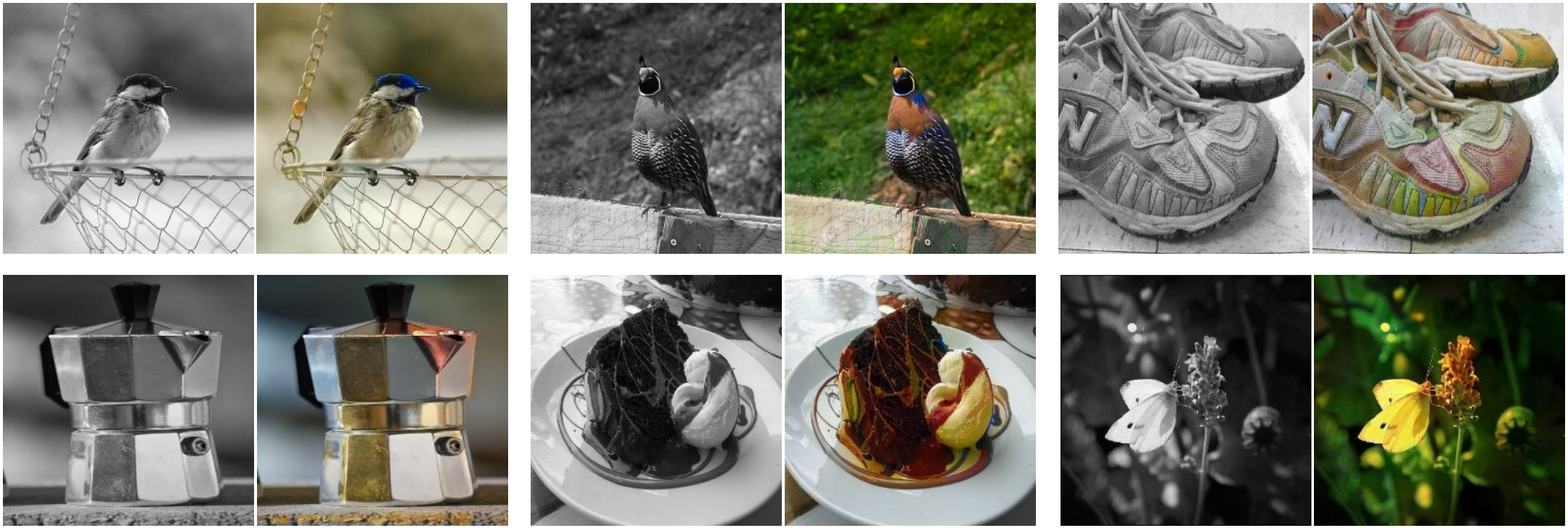
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Conditional GANs (cGAN)



BW → Color

Input Output Input Output Input Output



Data from [Russakovsky et al. 2015]

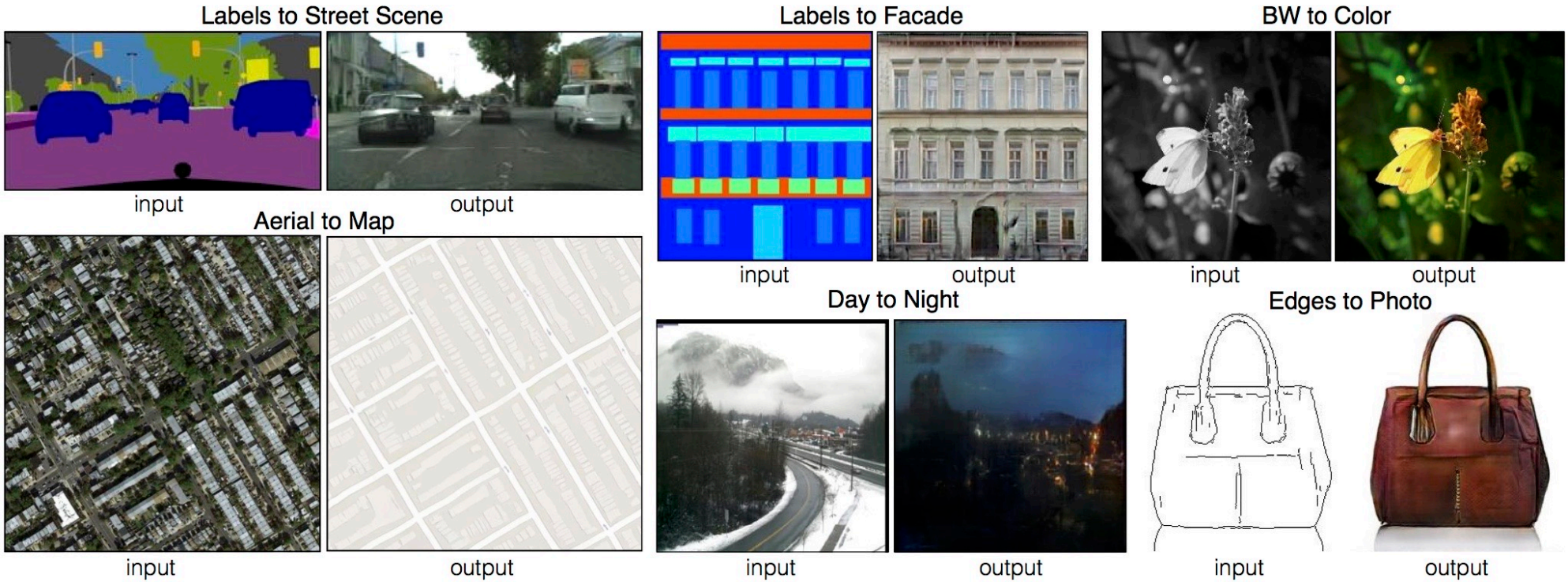
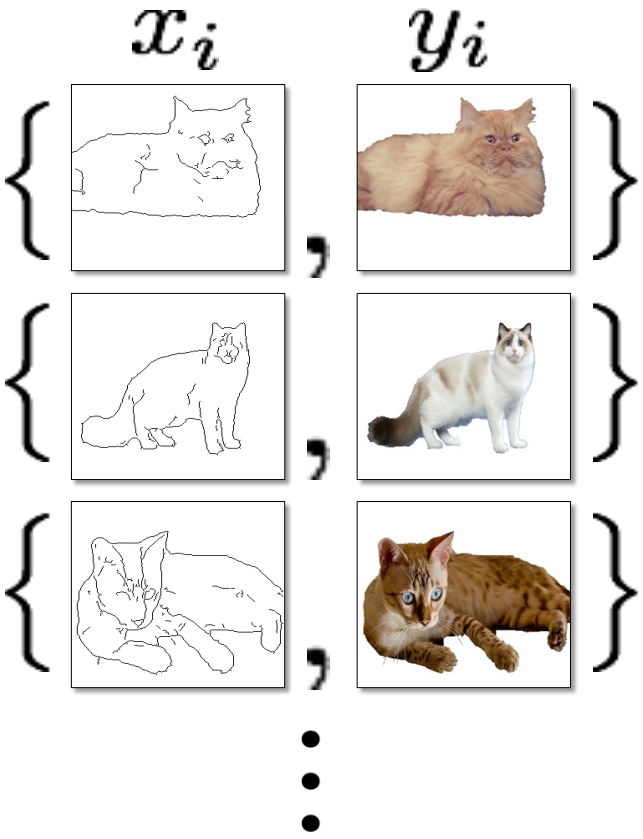
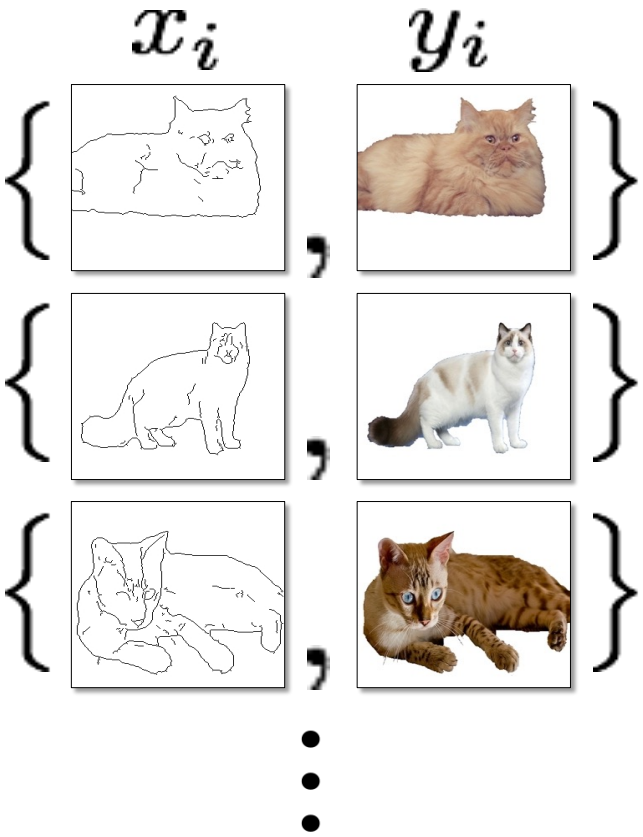


Image-to-Image Translation with Conditional Adversarial Networks
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros
CVPR, 2017.

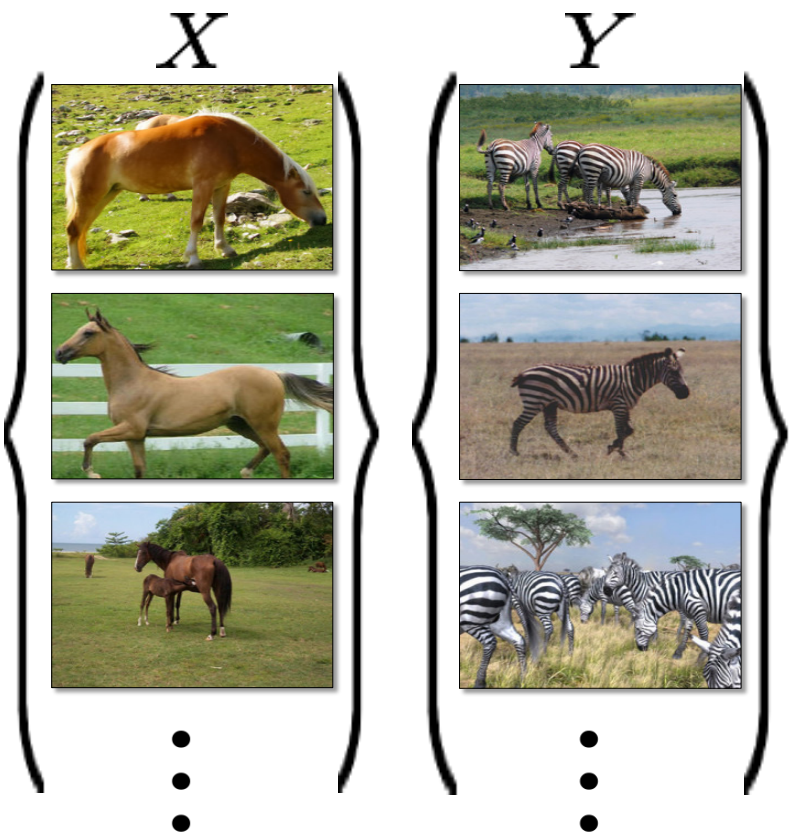
Paired data



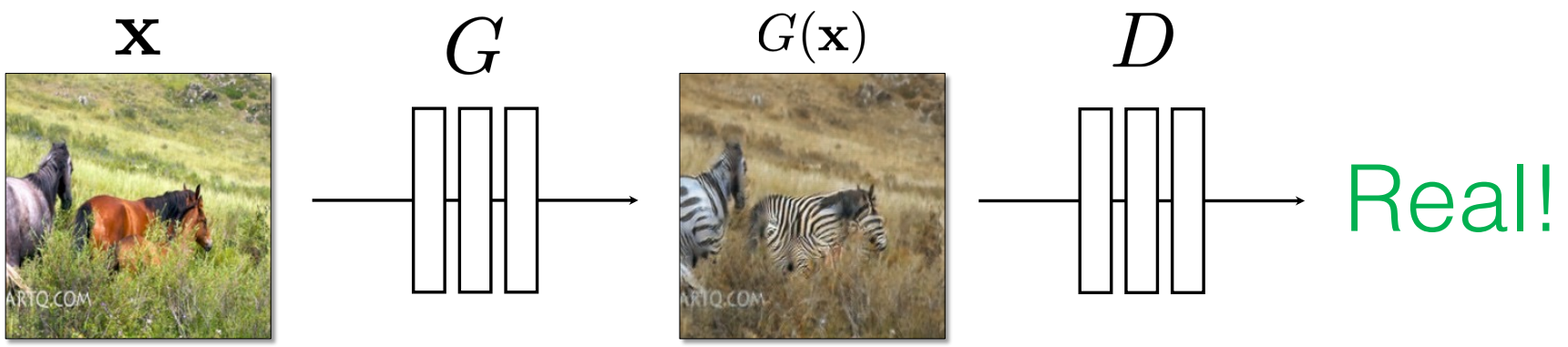
Paired data

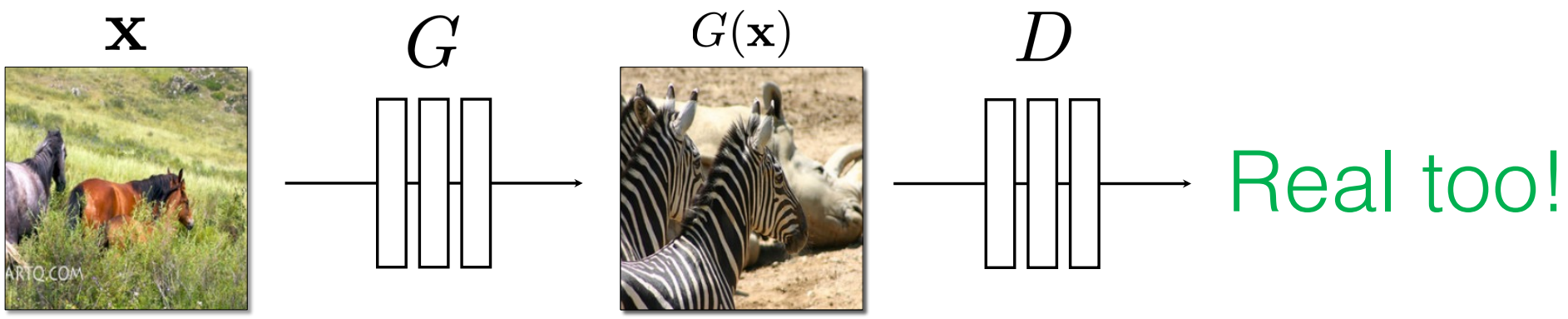


Unpaired data



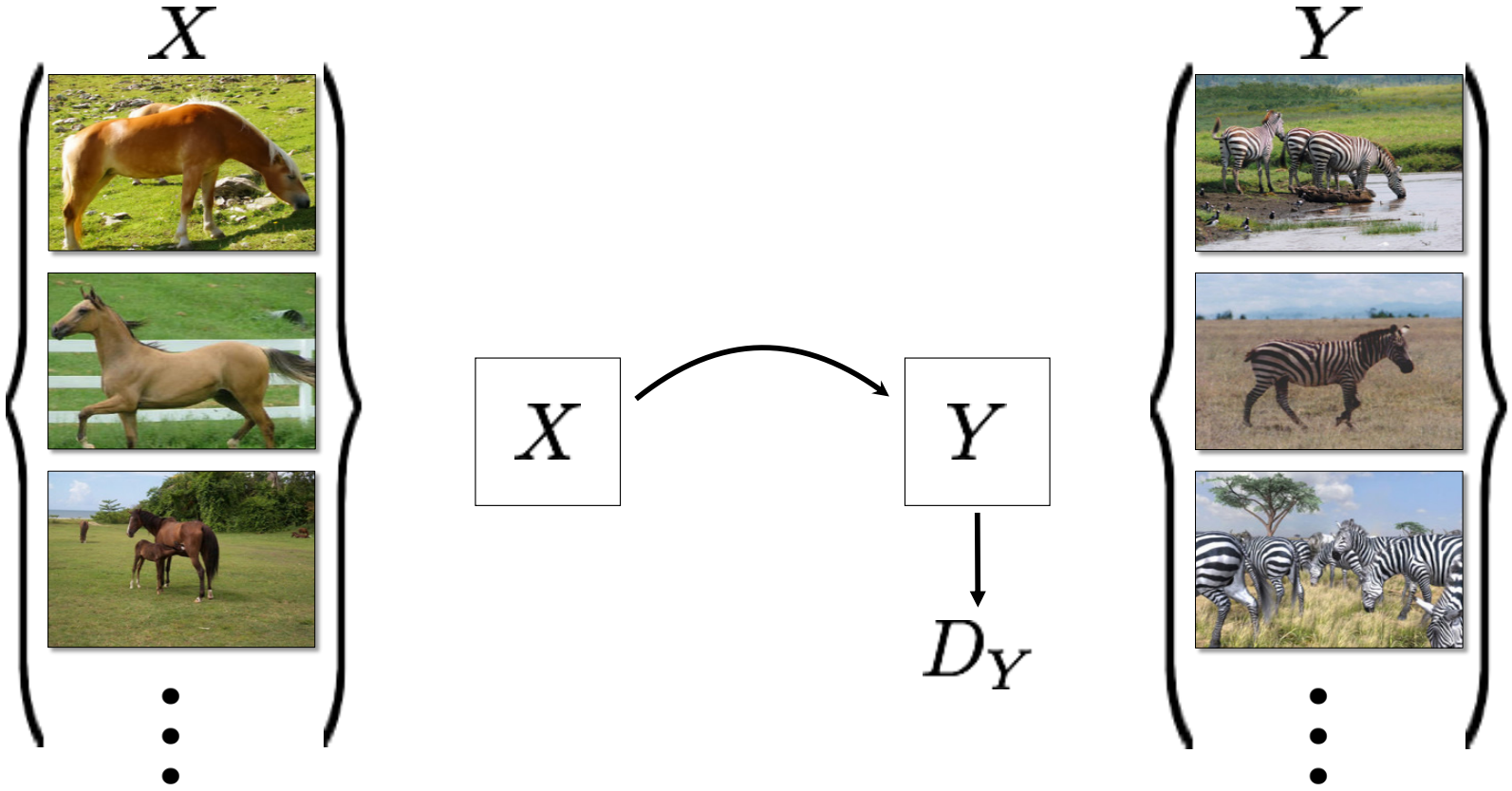
adapted from MIT 6.819/6.869 - Bill Freeman, Antonio Torralba, Phillip Isola


$\mathbf{x}$
G
$G(\mathbf{x})$
D
Real!



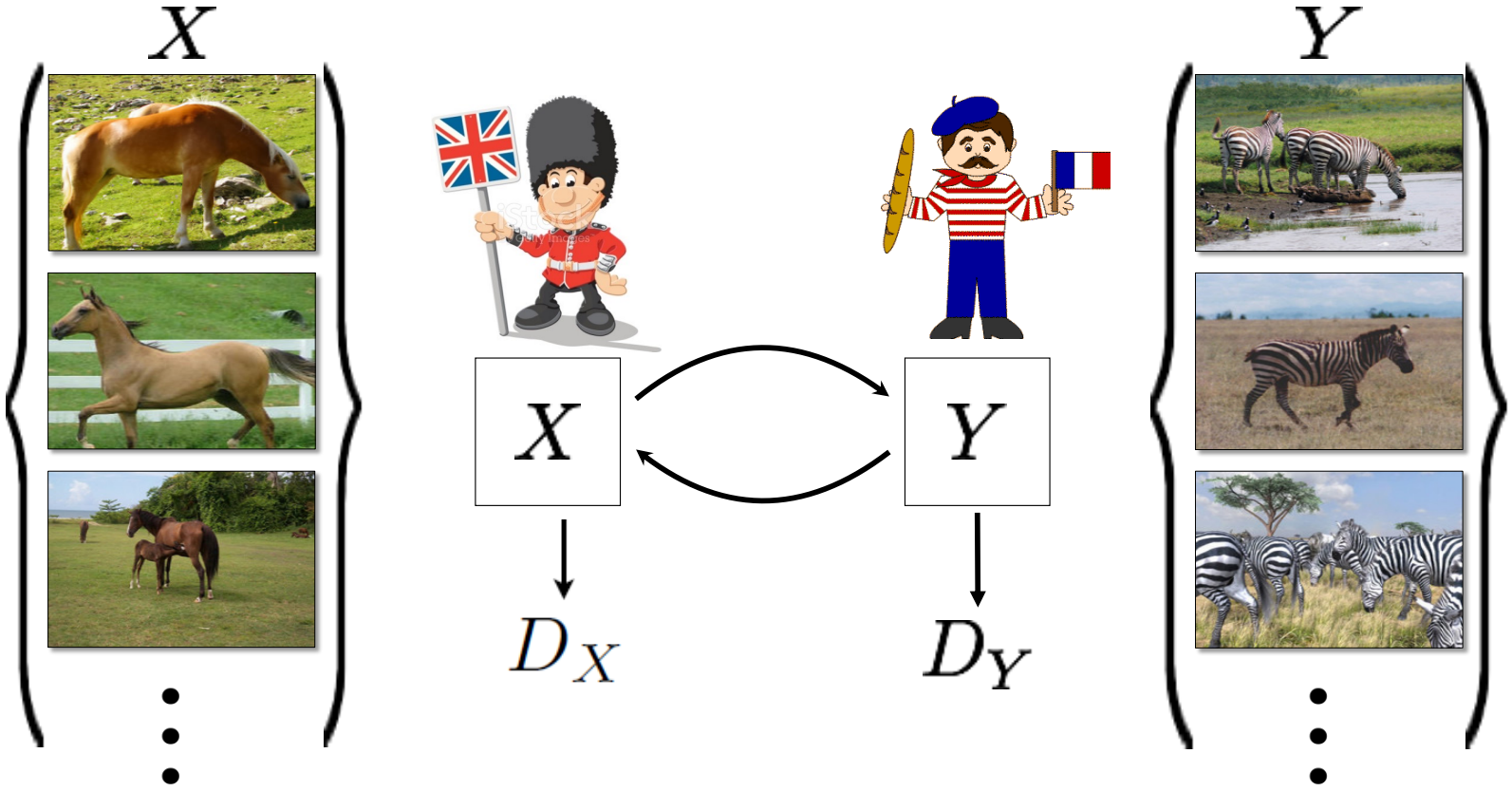
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks



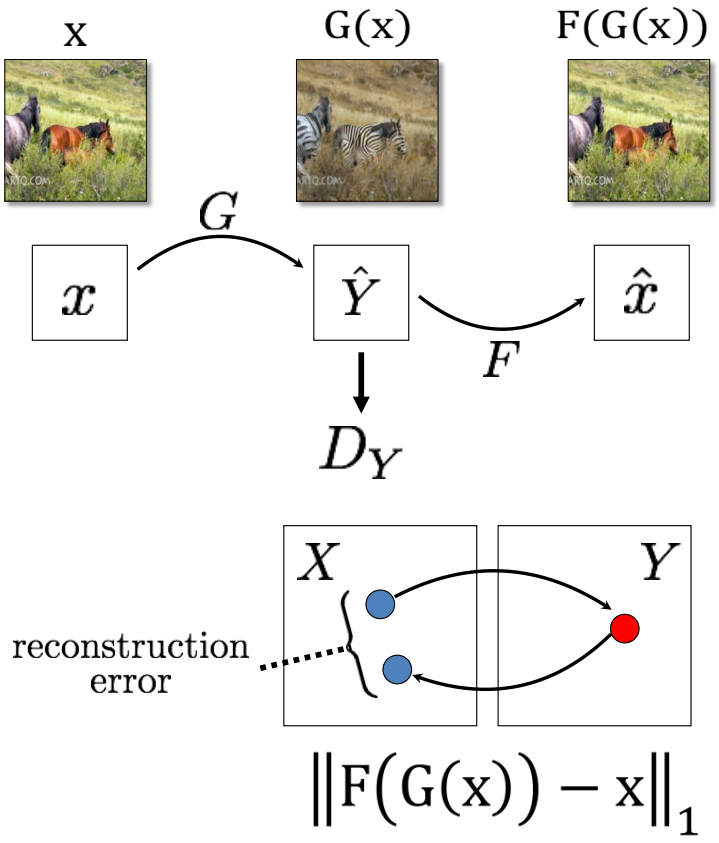
[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks

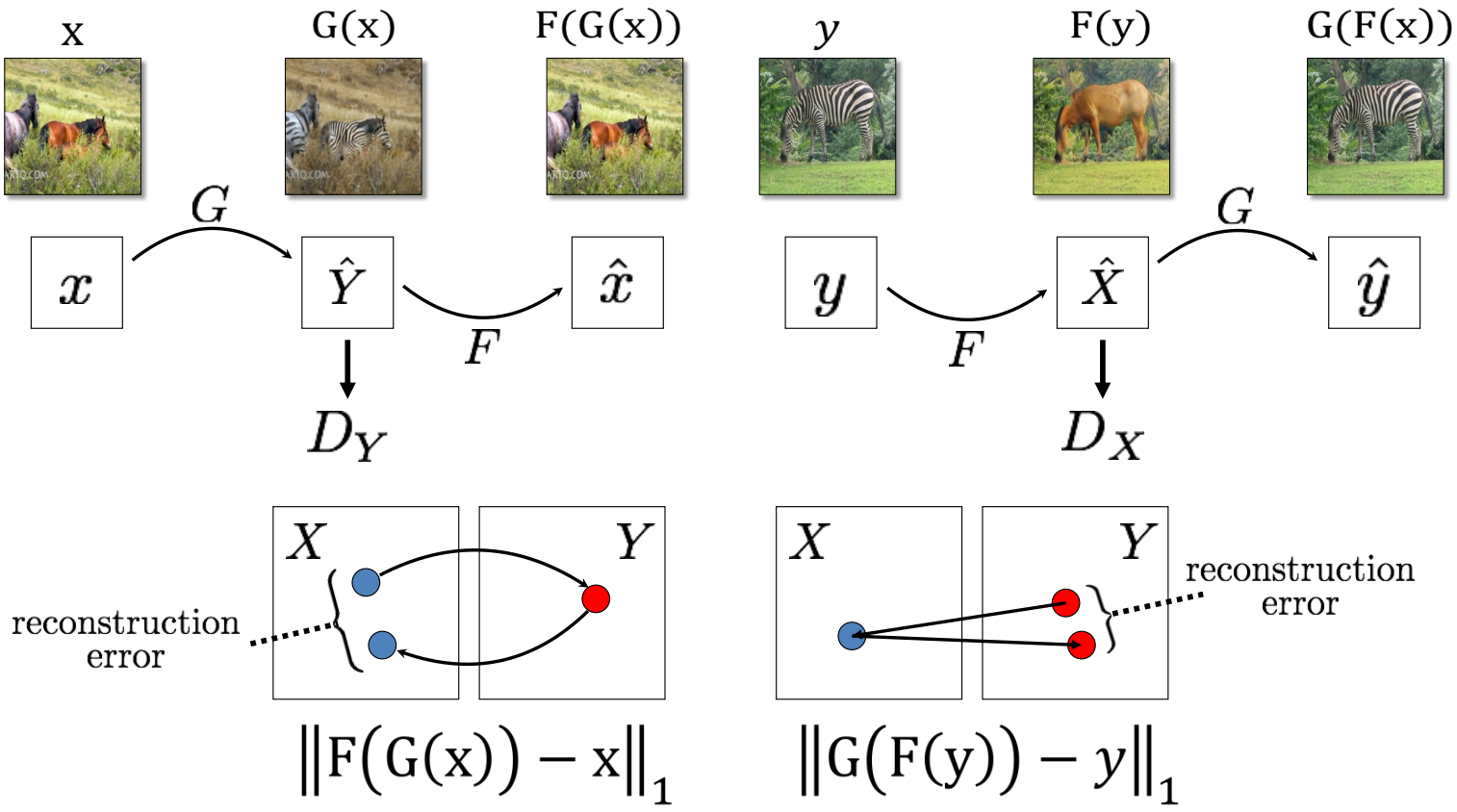


adapted from MIT 6.819/6.869 - Bill Freeman, Antonio Torralba, Phillip Isola

Cycle Consistency Loss



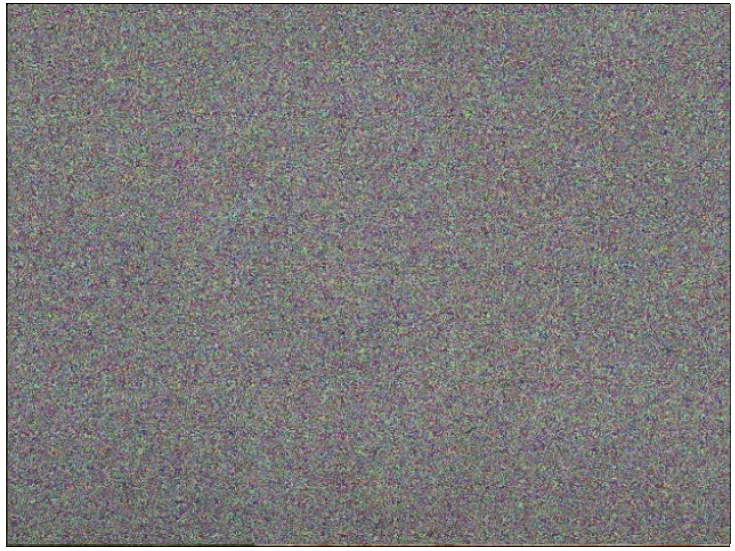
Cycle Consistency Loss




Input
Monet
Van Gogh
Cezanne
Ukiyo-e

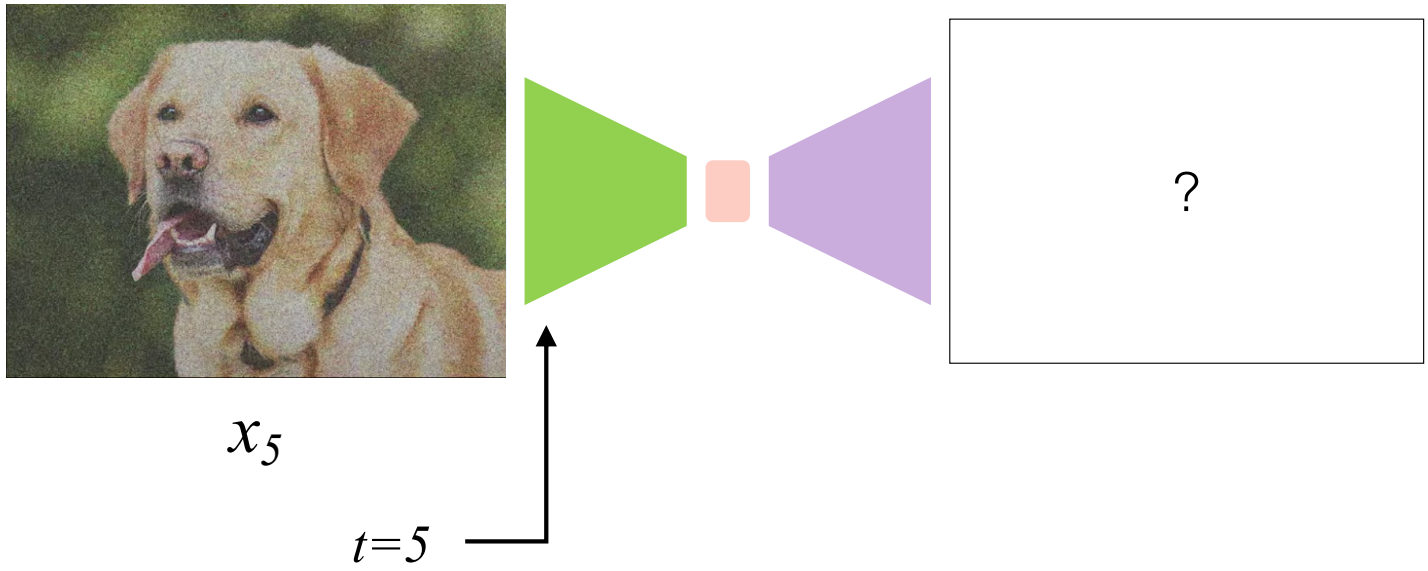
Monet’s paintings -> photos

Diffusion Models

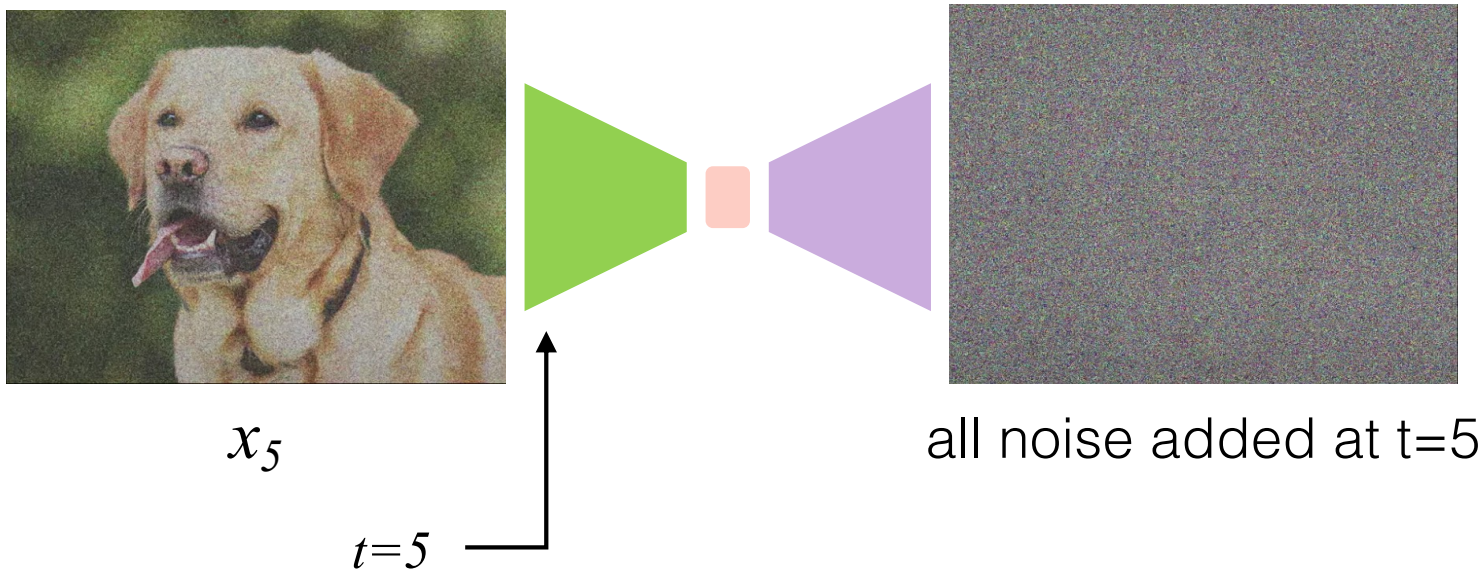


Diffusion Models

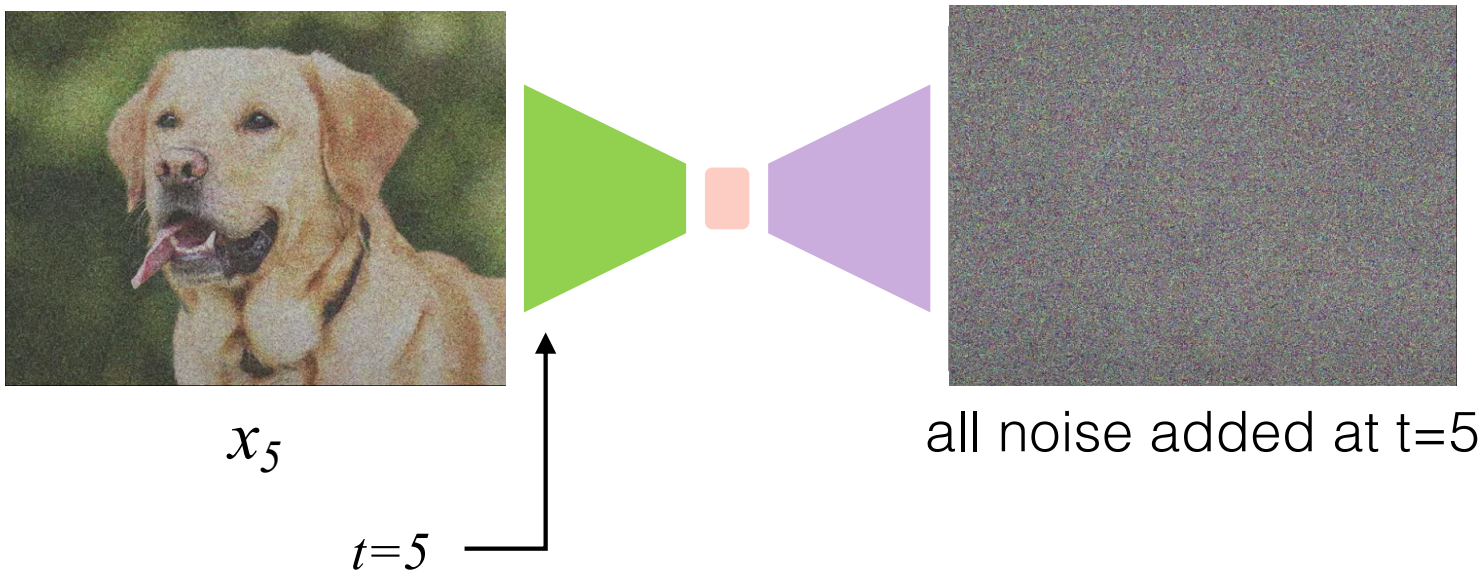
Diffusion Models



Diffusion Models



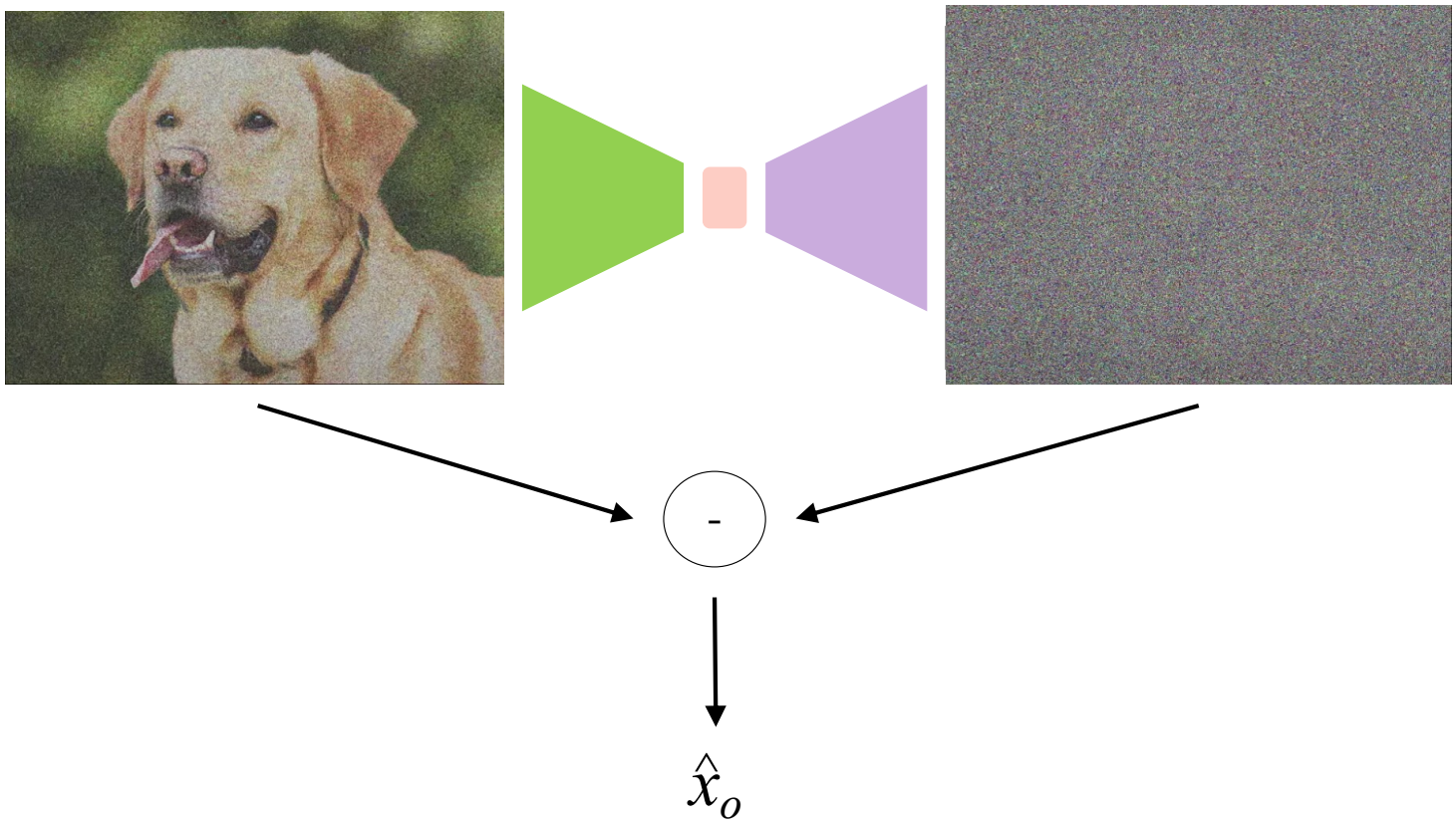
Diffusion Models



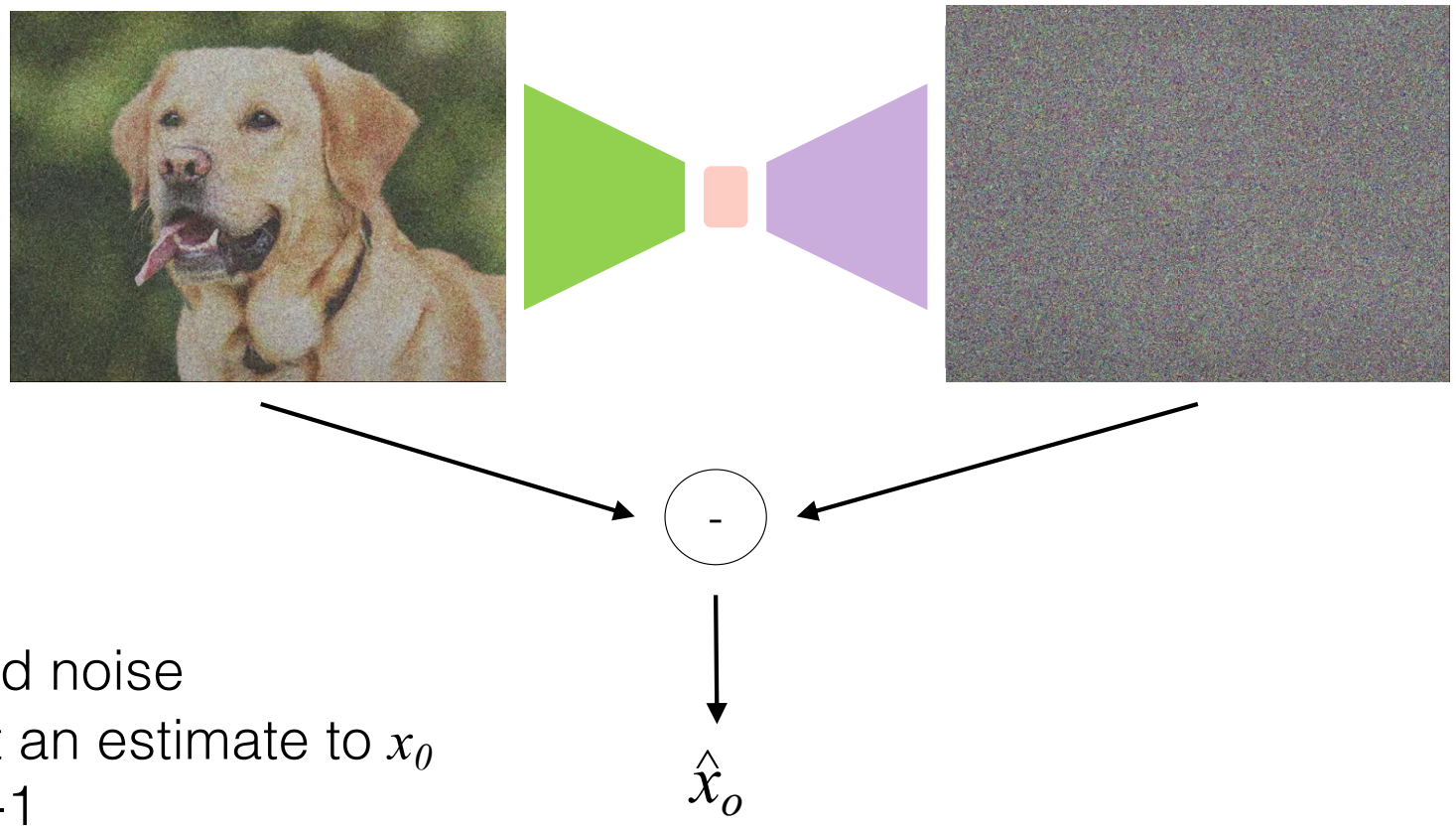
During training:

- Pick a random source image
- Pick a random time step
- Add noise based on the noise schedule
- Update the network based on the prediction

Diffusion Models



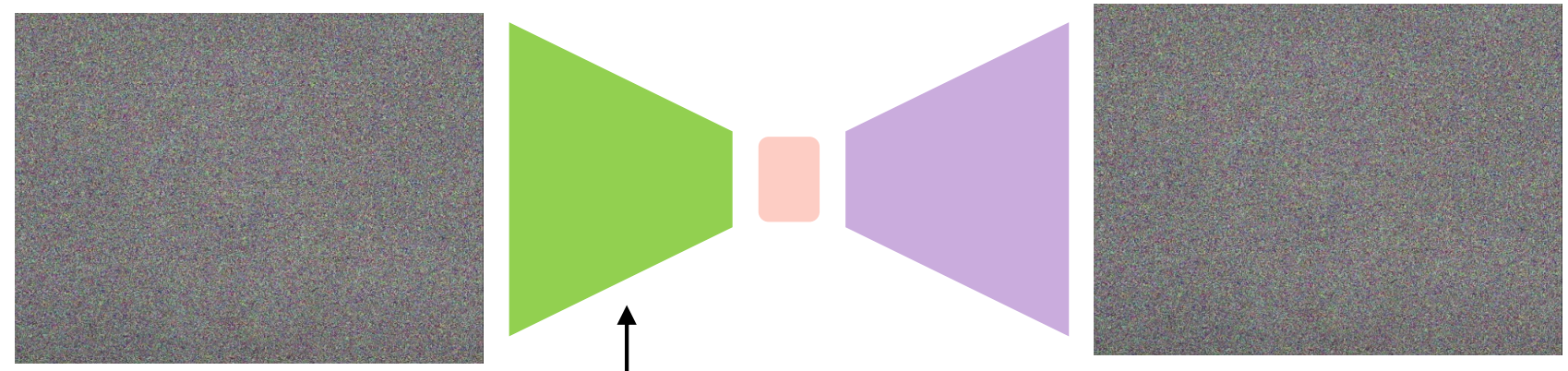
Diffusion Models



During inference:

- Start at t=T
- Predict the added noise
- Subtract and get an estimate to x_0
- Add noise until t-1
- Repeat the process until t=0

Diffusion Models



Text input

Language model
(e.g. GPT-3, BERT)

Embeddings

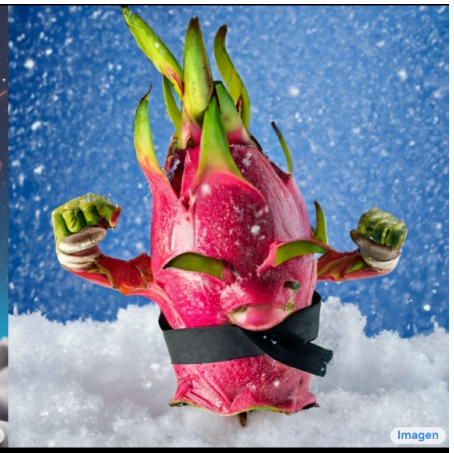
Repeat same
process as before

Diffusion Models

- DALL-E 2
  https://openai.com/dall-e-2/
- Imagen
  https://imagen.research.google/
- Stable Diffusion
  https://stability.ai/blog/stable-diffusion-public-release

An astronaut riding a horse in a photorealistic style

A brain riding a rocketship heading towards the moon.



A dragon fruit wearing karate belt in the snow.